Titre: Title:	Optimisation stochastique d'horaires de personnel
Auteur: Author:	Rémi Pacqueau
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Pacqueau, R. (2011). Optimisation stochastique d'horaires de personnel [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/595/

UNIVERSITÉ DE MONTRÉAL

OPTIMISATION STOCHASTIQUE D'HORAIRES DE PERSONNEL

RÉMI PACQUEAU
DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE
MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES ET GÉNIE INDUSTRIEL)
JUIN 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

OPTIMISATION STOCHASTIQUE D'HORAIRES DE PERSONNEL

présenté par : M. PACQUEAU Rémi, Ing.
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury constitué de :

M. DESAULNIERS Guy, Ph.D., président.
M. SOUMIS François, Ph.D., membre et directeur de recherche.
M. GENDREAU Michel, Ph.D., membre.

À tous mes amis du GERAD

Remerciements

Je tiens en premier lieu à remercier François Soumis, pour avoir accepté de diriger ma maîtrise et pour son aide régulière. C'est en grande partie grâce à lui que je garderai un excellent souvenir de mon passage à Montréal. Je remercie également l'École Polytechnique de Montréal ainsi que l'entreprise KRONOS, qui ont participé au financement de mes études.

Je tiens également à remercier tout le personnel du GERAD et du département MAGI, pour leur gentillesse, leur bonne humeur et leur disponibilité.

Je remercie Antoine Legrain pour son aide et ses précieux conseils.

Je remercie également tous les habitués du tarot de midi, pour les bons moments passés avec eux, entre deux articles à lire ou deux simulations à programmer.

Je remercie chaleureusement tous mes amis du GERAD et de Montréal, avec lesquels j'ai passé un séjour inoubliable au Québec, et qui ont souvent servi de cobayes pour mes présentations ou de relecteurs pour mes rapports.

Je remercie enfin tous ceux que j'aurai pu oublier et qui ont contribué de près ou de loin à cette maîtrise, en particulier tous les professeurs que j'ai pu avoir et qui m'ont permis d'aller toujours plus loin dans les études.

Résumé

La recherche opérationnelle peut intervenir à différentes phases de l'optimisation d'horaires : construction des jours de travail et de repos, construction des quarts de travail, allocation des quarts et des tâches aux employés. Lorsque l'information sur la demande en employés est imparfaite, on effectue ces étapes en différé : construction des jours de travail et de repos avec une demande très grossière quelques mois à l'avance, construction des quarts quelques semaines à l'avance avec une demande prévisionnelle plus fine, allocation des quarts et « recours » (c'est-à-dire emploi de temps partiels, modification des pauses et heures supplémentaires) le jour « J », une fois que l'on dispose de la demande précise.

Dans cette maîtrise, nous nous intéressons à une méthode différente de construction des quarts. Au lieu de prendre en compte seulement une demande prévisionnelle, on prendra en compte dans l'optimisation la distribution de cette demande aléatoire. L'objectif sera alors de minimiser *l'espérance* du coût total, *i.e.* du coût des quarts réguliers et du recours. Au prix d'un alourdissement du problème, on arrivera à une solution meilleure en pratique.

Après avoir formulé le problème, nous développons une heuristique basée sur une méthode célèbre en programmation stochastique : la méthode *L-shaped*. Notre heuristique est efficace : les pertes d'optimalité sont faibles, et le temps de résolution est raisonnable : moins de 15 minutes en général pour 500 scénarios (10 millions de variables), à comparer aux presque 5 heures nécessaires à `CPLEX` pour résoudre simplement la relaxation linéaire du problème.

Nous étudions alors sur différents jeux de données le gain apporté par cette approche stochastique par rapport à l'approche déterministe. Nous constatons que, dans les cas où la variance des jeux de données est raisonnable, les gains sont de 1 à 2 % du coût total. Lorsque la variance est faible, aucun gain n'est à constater. Pour des cas de très grande variance, le gain peut aller jusqu'à 15%. Nous en concluons que notre approche présente un intérêt certain, mais que ce dernier dépend grandement du type de demande en employés présentée.

Abstract

There are different steps in workforce planning where operations research can be helpful : building days in/off schedules, building work shifts, and allocating shifts and tasks to employees. When the employee demand is not known precisely in advance, those steps have to be done at different moments : building days in/off schedules several months in advance with a draft forecast demand, building shifts several weeks in advance with a more precise but still forecast demand, and finally allocating shifts and taking a "recourse" (hiring part-time employees, asking some employees to do over-time, building the pause schedule) on " D-day ", once the exact demand is known.

In this master's thesis we study a different way of building shifts. Instead of making the shift construction based only on a forecast demand, we use the demand's complete probability distribution. Thus, the goal becomes to minimize the *expected value* of the total cost, *i.e.* the cost of the regular shifts and the cost of the recourse. By increasing the problem's size, we compute a better solution than the one we would have using the classical, deterministic method.

After having formulated the problem, we develop a heuristic based on a well-known stochastic programming algorithm : the *L-shaped* method. Our heuristic turns out to be very efficient : optimality losses are small, and the computational times are reasonable : most of the time less than 15 minutes for a 500-scenario problem (10 million variables), where `CPLEX` takes a little less than 5 hours only to solve the linear relaxation of the problem.

We then study on different data sets the savings we manage to get compared to a deterministic approach. In cases where the demand's variance is reasonable, savings from 1 to 2 % of the overall cost can be obtained. When the variance is small, no significant savings can be expected. For cases with important variances, savings can be up to 15 %. We conclude that our approach is worth of interest, but that savings dramatically depend on the kind of employee demand we are dealing with.

Table des matières

Liste des tableaux

Liste des figures

Liste des sigles et abréviations

MIP	Mixed Integer Programming (*Programmation linéaire mixte*)
LP	Linear Programming (*Programmation linéaire*)
IP	Integer Programming (*Programmation en nombres entiers*)
EV	Expected Value problem (*Problème en valeur moyenne*)
EEV	Expected result of using the EV solution (*Espérance de la solution moyenne*)
VSS	Value of the Stochastic Solution (*Valeur de la solution stochastique*)
RP	Recourse Problem (*Problème avec recours*)
WS	Wait and See
PSNE	Programme Stochastique en Nombres Entiers

Chapitre 1

INTRODUCTION

Les problèmes de gestion de personnel sont courants en recherche opérationnelle. Lorsque la demande en employés est stochastique (donc incertaine), l'optimisation d'horaires de personnel se réalise en plusieurs étapes, décrites dans le livre de Baptiste *et al.* (2005). Nous décrivons ici le cas d'employés avec tâches *interruptibles* : un employé peut en remplacer un autre lorsque celui-ci est en fin de quart ou part pour une pause.

Construction des cycles : quelques mois à l'avance, il faut être en mesure de dire aux employés quels jours ils vont travailler, et quels jours ils seront en congé. Ils doivent en effet pouvoir planifier leur temps personnel. On se base sur une estimation grossière de la demande journalière pour résoudre ce problème.

Construction des quarts : une fois que chaque employé connaît ses jours de travail et de repos, l'employeur doit construire les différents quarts de travail de chaque journée. Un quart est constitué d'un instant de début, d'un instant de fin, et d'une pause. Construire les quarts revient à déterminer *combien* d'employés assigner à chaque type de quart. Par exemple : 3 employés sur le quart 8h-16h avec pause entre 12h et 12h30, 4 employés sur le quart 10h30-18h30 avec une pause entre 14h et 14h30. On peut également attribuer à chaque quart une fenêtre de pause seulement, et décider plus tard quand chaque pause sera prise. La construction des quarts s'effectue quelques semaines en avance, afin de bénéficier d'une information plus précise sur la demande en employés du jour J.

Assignation des quarts : lorsque l'on sait combien d'employés attribuer à chaque quart, il faut effectuer l'assignation de ces quarts aux employés disponibles pendant cette période (la disponibilité ayant été décidée lors de la construction de cycles). On décide également d'un *recours* à prendre, afin de satisfaire la demande (que l'on suppose maintenant connue) : embaucher des personnes à temps partiel, demander

à certains employés de faire des heures supplémentaires, éventuellement décider de l'emplacement exact des pauses.

Assignation des tâches : dans le cas d'un problème où plusieurs tâches sont à réaliser en parallèle, il faut ensuite assigner une activité à chaque employé. Cela ne nous intéressera pas ici.

La méthode classique est de suivre séquentiellement ces étapes : construction des cycles en utilisant une demande grossière, construction des quarts avec une demande prévisionnelle plus précise, assignation des quarts / des tâches et recours le jour J ou très peu avant. Chaque problème est relativement simple et rapide à résoudre.

Dans cette maîtrise, nous nous intéresserons à une autre manière de construire les quarts et le recours. Au lieu d'utiliser une demande prévisionnelle pour la construction des quarts puis d'adapter le recours à chaque réalisation de l'aléa, nous nous baserons, pour la construction des quarts, sur la distribution stochastique de cette demande. Concrètement, on considérera que l'aléa peut se décomposer en N (très grand) scénarios représentant chaque courbe de demande possible pour une journée. Notre but sera alors de construire les quarts réguliers qui minimisent *l'espérance* du coût total (quarts réguliers *et* recours), sur toutes les réalisations possibles de l'aléa.

Cela correspondra en fait à considérer non plus une moyenne, mais une distribution de probabilité. En « gagnant » ainsi en information, on sera à même de proposer une solution moins coûteuse, au prix d'un problème plus lourd à résoudre, appelé *problème stochastique*.

Plan du mémoire : nous commencerons tout d'abord par présenter en détail notre problème, étudier la classe générale des problèmes stochastiques et présenter une méthode classique de résolution dans le cas continu : la méthode *L-shaped*, basée sur la décomposition de Benders (1962). Nous dresserons également un état de l'art de la résolution des problèmes stochastiques en nombres entiers. Ensuite, nous développerons, implémenterons, analyserons et testerons une heuristique de résolution pour notre problème. Enfin, nous discuterons de l'intérêt ou non d'une telle approche.

Chapitre 2

REVUE DE LITTÉRATURE

Le livre de Baptiste *et al.* (2005) décrit les différentes approches adoptées en recherche opérationnelle concernant la gestion d'horaires de personnel. Un article classique de Dantzig (1954) présente une des premières formulations du problème de construction de quarts comme un problème de recouvrement.

De nouvelles approches utilisent d'autre formulations, dites « implicites », comme celles de Aykin (1996) et de Bechtold et Jacobs (1990). L'idée est de ne plus expliciter chaque quart possible (*i.e.* donner une heure de début, une heure de fin, et une période précise de pause), mais d'attribuer à chaque quart une fenêtre de pause, et de décider ensuite comment seront prises les pauses. En plus de diminuer le nombre de variables, cela permet de séparer le problème de construction de quarts et celui d'affectation des pauses : deux instants de décision sont donc possibles. Nous présenterons la formulation de Aykin (1996) et l'adapterons au cas stochastique dans le chapitre 3.

Bard *et al.* (2007) ont déjà étudié l'approche stochastique du problème de construction de quarts pour la poste américaine, avec des réglementations qui lui sont spécifiques. Comme nous allons le faire, ils décomposent le problème en deux instants de décision : construction des quarts réguliers puis recours. Après une étude quantitative, les auteurs constatent qu'une approche stochastique leur permet d'économiser de l'ordre de 5 % du coût total : c'est un résultat encourageant. En revanche, ils se limitent à des problèmes comportant 3 scénarios (maximum 6500 variables et 3300 contraintes), et résolvent la relaxation linéaire de leur problème uniquement par l'algorithme du simplexe (une heuristique s'ensuit pour déterminer la solution en nombres entiers). La notion de scénarios sera expliquée lors de la présentation de la programmation stochastique, au chapitre 3.

Notre ambition sera de résoudre des problèmes de taille beaucoup plus importante (nous verrons que notre problème final comptera 10 millions de variables), et par conséquent de développer une méthode de résolution adaptée. Le livre de Birge et Louveaux (1997) expose les méthodes classiques pour résoudre de tels problèmes (en linéaire ou en nombres entiers).

Une méthode célèbre est la méthode *L-shaped*, basée sur la décomposition de Benders (1962). Cette méthode est efficace pour résoudre de gros problèmes stochastiques linéaires. Il en existe une variante en nombres entiers, mais cette dernière est peu utilisée.

Un état de l'art récent de la programmation stochastique en nombres entiers a été dressé par Sherali et Zhu (2009). Ainsi, Ahmed *et al.* (2004) reformulent le problème en « tender variables » pour éliminer certaines discontinuités dans le coût du recours. Caroe et Schultz (1999) résolvent eux le problème par relaxation lagrangienne. Deux approches plus originales sont également proposées : Kong *et al.* (2006) utilisent la super-additivité duale pour énumérer explicitement toutes les valeurs possibles du recours, et Hemmecke et Schultz (2003) utilisent un algorithme d'augmentation semblable à ce que l'on peut trouver en théorie des graphes.

Nous ne développons dans cette revue de littérature aucune des méthodes précédemment citées, car elles feront l'objet d'un traitement en profondeur dans ce rapport de maîtrise. La méthode L-shaped sera ainsi traitée dans le chapitre 3. Les différentes pistes déjà explorées pour la résolution de problèmes stochastiques en nombres entiers seront présentées en détail dans le chapite 4. Nous verrons que le principal inconvénient de ces méthodes est qu'elles ne sont efficaces en général que pour des formes très spécifiques de problèmes : énormément de scénarios mais très peu de contraintes primaires, par exemple.

Chapitre 3

PREMIÈRE FORMULATION, MÉTHODES CLASSIQUES DE RÉSOLUTION

3.1 Position du problème et formulation

Le problème porte sur la gestion d'horaires de personnel dans un contexte incertain. Considérons par exemple un supermarché, pour lequel on souhaite construire un outil de gestion des horaires des caissiers. En plus de varier au cours de la journée, la demande varie également d'un jour à l'autre, tout cela de manière aléatoire (météo, trafic routier, ...). On supposera par la suite que l'ensemble des réalisations de l'aléa est de cardinalité finie. Chaque réalisation est un « scénario » comportant la demande en employés sur une journée (discrétisée en 96 périodes de 15 minutes).

A supposer que le gérant du supermarché connaisse précisément la demande seulement d'un jour à l'autre, la solution optimale serait d'attendre la dernière minute pour attribuer à chaque employé ses horaires de travail et ses jours de congé. Cette solution est appelée, dans le langage de la programmation stochastique, la solution « wait and see ». Elle n'est toutefois pas réaliste, car il faut prévenir les employés de leurs jours de travail et de leurs horaires approximatifs beaucoup plus en avance.

Une autre approche consisterait alors à prendre comme donnée de notre problème d'optimisation la demande moyenne observée (ou une demande prévisionnelle), et d'allouer des quarts de travail afin de respecter cette demande. On effectuerait ensuite un recours propre à chaque scénario afin de répondre à la demande effective. Cette solution est appelée la solution du problème *moyen* (ou encore problème *déterministe*). Elle est relativement rapide à calculer, mais sous-optimale compte tenu de l'infor-

mation que l'on a. En effet, on a à notre disposition la distribution d'une variable aléatoire et on n'en exploite que sa moyenne.

La troisième approche, que nous avons étudiée, est appelée la solution « here and now » (ou *stochastique*). Elle consiste également à échelonner les prises de décision en deux temps. On planifie longtemps en avance les quarts réguliers et on prévoit, pour chaque scénario, un *recours* visant à répondre à la demande du scénario en question.

Néanmoins, au lieu d'utiliser uniquement la moyenne de la demande pour sélectionner les quarts réguliers à attribuer, le problème consistera à trouver une décision primaire qui minimise *l'espérance*, sur toutes les réalisations possibles de l'aléa, du coût total nécessaire (décision primaire *et* recours) afin de garantir une couverture de la demande. Alors que décision primaire et décision secondaire n'étaient pas couplées dans la solution moyenne, elles le sont dans l'approche « here and now », qui de ce fait utilise l'intégralité de l'information mise à notre disposition.

Ce problème combine deux difficultés majeures :

1. La programmation en nombres entiers
2. La programmation stochastique

Puisque le nombre de scénarios est fini, nous verrons qu'il est possible de reformuler notre problème dans une forme linéaire dite « déterministe équivalente », à priori résoluble par n'importe quel solveur de programmation linéaire mixte (MIP). Néanmoins, l'immense taille du problème (en partie due à la multiplicité des scénarios) rend en pratique la résolution par un solveur MIP classique impossible.

3.1.1 Notations

Nous adaptons ici une formulation « implicite », introduite par de Aykin (1996). La décision primaire consiste à construire des quarts à réguliers (en général d'une durée de 8 heures). Ces quarts disposent chacun d'une fenêtre de temps où une pause devra obligatoirement être prise. Une fois que la demande est connue, la décision secondaire consiste à décider de l'emplacement des pauses et éventuellement de rajouter des heures supplémentaires (à la fin des quarts réguliers seulement, sans modification des

pauses) ou à créer des quarts à temps partiel (sans pause), dans le but de couvrir la demande spécifique du scénario. Les tableaux 3.1-3.3 résument toutes les notations.

Notation	Signification
P	Ensemble des périodes
J	Ensemble des quarts réguliers
JP	Ensemble des quarts à temps partiel
H	Ensemble des heures supplémentaires
K	Ensemble des pauses
Ω	Ensemble des évènements aléatoires

Tableau 3.1 Ensembles du problème

Remarque 3.1.1. *En pratique, on s'assurera que la taille de ces ensembles n'est pas trop élevée, afin de pouvoir explicitement énumérer leurs éléments et éviter ainsi d'avoir recours à de la génération de colonnes, qui alourdirait un problème déjà complexe.*

Notation	Signification
S_j	Nombres d'employés assignés au quart $j \in J$
SP_j^ω	Nombres d'employés assignés au quart partiel $j \in JP$ pour la réalisation $\omega \in \Omega$
SH_h^ω	Nombre de fois où l'heure supplémentaire $h \in H$ est attribuée pour la réalisation $\omega \in \Omega$
B_k^ω	Nombre d'employés à temps plein utilisant la pause $k \in K$ dans la réalisation $\omega \in \Omega$
$X_{j,k}^\omega$	Nombre d'employés assignés au quart j et utilisant la pause k dans la réalisation $\omega \in \Omega$
$XH_{j,h}^\omega$	Nombre d'employés assignés au quart j et effectuant l'heure supplémentaire h dans la réalisation $\omega \in \Omega$

Tableau 3.2 Variables de décision du problème

3.1.2 Formulation

Nous pouvons maintenant présenter la première formulation (P1) de notre problème. L'objectif est séparé entre une partie primaire et une partie secondaire (qui n'est rien d'autre que l'espérance du coût du recours).

Notation	Signification
c_j	Coût du quart $j \in J$
cp_j	Coût du quart à temps partiel $j \in JP$
cs_h	Coût du créneau d'heures supplémentaires $h \in H$
p_ω	Probabilité de réalisation de l'évènement $\omega \in \Omega$
D_p^ω	Demande à la période $p \in P$ pour la réalisation $\omega \in \Omega$
A_{jp}	Vaut 1 si le quart $j \in J$ couvre la période $p \in P$, 0 sinon
AP_{jp}	Vaut 1 si le quart partiel $j \in JP$ couvre la période $p \in P$, 0 sinon
AK_{kp}	Vaut 1 si la pause $k \in K$ couvre la période $p \in P$, 0 sinon
AH_{hp}	Vaut 1 si le bloc d'heures supplémentaires $h \in H$ couvre la période $p \in P$, 0 sinon
Q_{jk}	Vaut 1 si la pause $k \in K$ est compatible avec le quart $j \in J$, 0 sinon
R_{jh}	Vaut 1 si le bloc d'heures supplémentaires $h \in H$ est compatible avec le quart $j \in J$, 0 sinon

TABLEAU 3.3 Constantes du problème

$$(P1) \quad \min \quad \sum_{j\in J} c_j S_j + \sum_{\omega\in\Omega} p_\omega \left[\sum_{j\in JP} cp_j SP_j^\omega + \sum_{h\in H} cs_h SH_h^\omega \right] \tag{3.1}$$

$$\text{slc} \quad \sum_{j\in J} A_{jp} S_j + \sum_{j\in JP} AP_{jp} SP_j^\omega - \sum_{k\in K} AK_{kp} B_k^\omega + \sum_{h\in H} AH_{hp} SH_h^\omega \geq D_p^\omega, \ \forall(\omega,p) \in (\Omega, P) \tag{3.2}$$

$$\sum_{k\in K} Q_{jk} X_{jk}^\omega - S_j = 0, \ \forall(\omega,j) \in (\Omega, J) \tag{3.3}$$

$$\sum_{j\in J} Q_{jk} X_{jk}^\omega - B_k^\omega = 0, \ \forall(\omega,k) \in (\Omega, K) \tag{3.4}$$

$$\sum_{h\in H} R_{jh} X_{jh}^\omega \leq S_j, \ \forall(\omega,j) \in (\Omega, J) \tag{3.5}$$

$$\sum_{j\in J} R_{jh} XH_{jh}^\omega - SH_h = 0, \ \forall(\omega,h) \in (\Omega, H) \tag{3.6}$$

$$S_j, \ SP_j^\omega, \ SH_h^\omega, \ B_k^\omega, \ X_{jk}^\omega, \quad XH_{jh}^\omega \in \mathbb{Z}^+, \ \forall(j,\omega,h,k) \in (\Omega, H, K) \tag{3.7}$$

Les contraintes (3.2) imposent la couverture de la demande pour toute période et pour tout scénario[1]. Les contraintes (3.3)-(3.4) imposent respectivement qu'il y ait une pause attribuée par quart régulier et que chaque pause attribuée soit compatible avec le quart auquel elle est associée. Les contraintes (3.5)-(3.6) assurent qu'il n'y ait pas plus d'heures supplémentaires associées à un quart j qu'il n'y a de personnes effectuant ledit quart, et que toutes les heures supplémentaires utilisées soient compatibles avec les quarts auxquels elles sont associées.

Profitons-en pour introduire le *problème secondaire* qui cherche, pour une décision maître fixée ν (donc S_j fixé pour tout j, noté S_j^ν), et pour une réalisation ω, le meilleur recours possible.

$$(SP1)_\omega^\nu \quad \min \quad \sum_{j\in JP} cp_j SP_j^\omega + \sum_{h\in H} cs_h SH_h^\omega \tag{3.8}$$

$$\text{slc} \quad \sum_{j\in J} A_{jp} S_j^\nu + \sum_{j\in JP} AP_{jp} SP_j^\omega - \sum_{k\in K} AK_{kp} B_k^\omega + \sum_{h\in H} AH_{hp} SH_h^\omega \geq D_p^\omega,\ \forall p \in P \tag{3.9}$$

$$\sum_{k\in K} Q_{jk} X_{jk}^\omega - S_j^\nu = 0,\ \forall j \in J \tag{3.10}$$

$$\sum_{j\in J} Q_{jk} X_{jk}^\omega - B_k^\omega = 0,\ \forall k \in K \tag{3.11}$$

$$\sum_{h\in H} R_{jh} X_{jh}^\omega \leq S_j^\nu,\ \forall j \in J \tag{3.12}$$

$$\sum_{j\in J} R_{jh} XH_{jh}^\omega - SH_h^\omega = 0,\ \forall h \in H \tag{3.13}$$

$$SP_j^\omega,\ SH_h^\omega,\ B_k^\omega,\ X_{jk}^\omega, \quad XH_{jh}^\omega \in \mathbb{Z}^+,\ \forall (j,h,k) \in (J,H,K) \tag{3.14}$$

Notons que la difficulté du problème vient du fait que les sous-problèmes et le problème maître sont dépendants (entre autres en raison des contraintes portant sur la demande), et qu'il est par conséquent impossible de les résoudre séparément. On voit immédiatement que la cardinalité de Ω aura une grande influence sur la taille du

[1] On introduira ultérieurement la possibilité de sous-couverture

problème, et sur sa résolution a priori.

S'ajoutant à cette difficulté, les contraintes d'intégrité font que contrairement au cas continu, la fonction de *recours* n'est plus convexe et linéaire par morceaux en les variables primaires de décision, phénomène qui, comme on le verra, facilitait la résolution des programmes stochastiques continus.

Nous présentons dans la suite de cette partie la classe des programmes linéaires stochastiques (en nombres entiers ou non), dont notre problème fait partie, ainsi que deux méthodes de résolution classiques (mais peu efficaces ou non adaptées) pour ces problèmes.

3.1.3 Programmes linéaires stochastiques

Tout au long de ce rapport, nous nous intéressons aux programmes stochastiques à deux étapes (*two-stage stochastic programs*). Ce problème est appelé problème stochastique avec *recours*. L'idée fondamentale est que l'on cherche à prendre une décision sur le futur en disposant d'une information imparfaite sur celui-ci. Cette incertitude est représentée par une famille d'évènements aléatoires Ω. Tout l'aléa est porté par une variable aléatoire $\xi(\omega) \in \Xi$.

La figure 3.1 décrit les différents temps de décision.

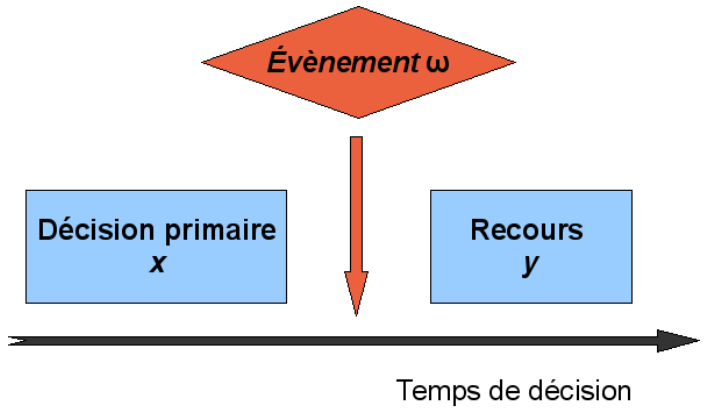


FIGURE 3.1 Les différents instants de décision

La variable x représente la décision *primaire* du problème. Elle est prise indépendemment de l'évènement $\omega \in \Omega$. Il s'agit en général d'une décision à long terme (choix d'implantation d'entrepôts, affectation grossière de personnel...), avant qu'une réalisation $\omega \in \Omega$ n'ait eu lieu . Les variables $y(\omega)$ représentent les *recours* possibles : pour chaque évènement ω, le vecteur $y(\omega)$ représente la meilleure réponse au problème pour x fixé. Les variables x et y sont liées par une contrainte dite de *non-anticipativité*.

La difficulté consiste alors à trouver les valeurs des variables primaires x qui minimisent *l'espérance* du coût total du problème. Cela se modélise sous la forme générale suivante :

$$\begin{aligned} \min_x z &= c^T x + \mathbb{E}_\xi \left(\min_y \left\{ q(\omega)^T y(\omega) \right\} \right) && (3.15) \\ Ax &= b && (3.16) \\ T(\omega)x + Wy(\omega) &= h(\omega),\ \forall \omega \in \Omega && (3.17) \\ x \geq 0 \quad &, \quad y \geq 0\ \forall \omega \in \Omega && (3.18) \end{aligned}$$

Le tableau 3.4 récapitule, pour toute la suite de l'exposé, les dimensions génériques du problème.

Notation	**Signification**
n_1	Nombre de variables primaires
n_2	Nombre de variables secondaires *par scénario*
m_1	Nombre de contraintes primaires
m_2	Nombre de contraintes secondaires (*idem*)
K	nombre de scénarios

TABLEAU 3.4 Taille du problème

Il est possible de réécrire 3.15 sous une forme condensée, appelée *forme déterministe équivalente* :

$$\begin{aligned} \min z &= c^T x + \mathcal{Q}(x) && (3.19) \\ Ax &= b && (3.20) \\ x &\geq 0 && (3.21) \end{aligned}$$

La fonction $x \mapsto \mathcal{Q}(x)$ est appelée *fonction de recours*. Elle représente l'espérance des meilleurs recours possibles pour x donné.

On a :

$$\mathcal{Q}(x) = \mathbb{E}_\xi \left(Q(x, \xi(\omega)) \right) \tag{3.22}$$

$$Q(x, \xi(\omega)) = \min_y \left\{ q(\omega)^T y | Wy = h(\omega) - T(\omega)x,\ y \geq 0 \right\} \tag{3.23}$$

Remarque 3.1.2. *Ce type de problème est appelé problème à* recours fixé*, car dans (3.17), W ne dépend pas de l'aléa.*

Nous allons maintenant donner une méthode classique de résolution des problèmes stochastiques avec recours, qui est essentiellement une reformulation de la décomposition de Benders (1962). Notre présentation reprend le plan et les notations utilisés dans le livre de Birge et Louveaux (1997).

3.2 La méthode L-Shaped

3.2.1 Principe

Basée sur la décomposition de Benders (1962), la méthode L-shaped est une méthode permettant de résoudre des problèmes stochastiques à deux étapes, dans le cas où ξ est une variable aléatoire discrète avec $|\Xi| < \infty$ et lorsqu'il n'y a pas de contrainte d'intégrité. Elle a été introduite par Van Slyke et Wets (1969). Même si elle n'est pas applicable en l'état pour notre problème (qui est en nombres entiers), son importance théorique fait que nous la présentons ici. Nous nous en servirons d'ailleurs afin de construire une heuristique de résolution pour notre problème.

Puisque nous supposons que $|\Xi|$ est fini, nous allons indicer par $k \in \{1, \ldots, K\}$ chacune de ses réalisations, de probabilité respective p^k. Nous notons (q^k, T^k, h^k) les paramètres correspondant à la réalisation k, et y^k le recours associé. Le problème (3.15)-(3.18) se reformule comme ceci :

$$
\begin{aligned}
(PI)\ \min_{x,y} z &= c^T x + \sum_{k=1}^{K} p^k \left[(q^k)^T y^k\right] && (3.24)\\
Ax &= b && (3.25)\\
T^k x + W y^k &= h^k,\ \forall k \in \{1,\dots,K\} && (3.26)\\
x \geq 0 \quad , &\quad y^k \geq 0\ \forall k \in \{1,\dots,K\} && (3.27)
\end{aligned}
$$

Les variables de décision correspondent donc au vecteur $(x, y_1, \dots, y_K)$, comprenant les décisions initiales x et les différents recours possibles en fonction de l'aléa. La forme (3.24)-(3.27) est celle d'un programme linéaire standard. Néanmoins, le grand nombre de variables et de contraintes rend dans la plupart des cas très longue, voire impossible, la résolution du problème par une méthode générique du type de l'algorithme du simplexe.

La matrice des contraintes de ce problème a tout de même une forme particulière qu'il est possible d'exploiter :

$$
\begin{pmatrix}
A & & & & 0\\
T_1 & W & & & \\
T_2 & 0 & W & & \\
\vdots & & & \dots & \\
T_K & & & & W
\end{pmatrix}
\tag{3.28}
$$

Cette structure creuse permet de décomposer le problème (PI) en différents sous-problèmes. Étant donné x^ν, on définit la famille de sous-problèmes suivante :

$$
\begin{aligned}
(SP^{k\nu})_{k\in\{1,\dots,K\}} \quad \min_{y^k} &\quad (q^k)^T y^k && (3.29)\\
W y^k &= h^k - T^k x^\nu && (3.30)\\
y^k &\geq 0 && (3.31)
\end{aligned}
$$

Le sous-problème dual s'écrit donc (dualisation partielle) :

$$(SD^{k\nu})_{k\in\{1,\dots,K\}} \qquad \max_{\lambda^k} \qquad (h^k - T^k x^\nu)^T \lambda^k \tag{3.32}$$

$$W^T \lambda^k \leq q^k \tag{3.33}$$

Notons $\Lambda^k = \{\lambda^k | W^T \lambda^k \leq q^k\}$, l'ensemble de réalisabilité du dual.

L'espace Λ^k étant une intersection d'un nombre fini d'hyperplans, c'est un polyèdre. On peut donc appliquer le théorème de Minkowsky-Weil et le décomposer comme somme d'un polytope P et d'un cône C. En notant $\pi^{kl},\ l \in \{1,\dots,p\}$ les points extrêmes de P et $\sigma^{kl},\ l \in \{1,\dots,q\}$ les directions extrêmes de C, on a ainsi :

$$\Lambda^k = \left\{ \sum_{l=1}^{p} \alpha^l \pi^{kl} + \sum_{l=1}^{q} \beta^l \sigma^{kl},\ \sum \alpha^l = 1,\ (\alpha,\beta) \geq 0 \right\} \tag{3.34}$$

Il est donc possible de reformuler $(SD)^{k\nu}$, pour tout $k \in \{1 \dots K\}$:

$$\max_{\lambda^k} z_d = (h^k - T^k x^\nu) \left(\sum_{l=1}^{p} \alpha^l \pi^{kl} + \sum_{l=1}^{q} \beta^l \sigma^{kl} \right) \tag{3.35}$$

$$\sum_{l=1}^{p} \alpha^l = 1 \tag{3.36}$$

$$\alpha, \beta \geq 0 \tag{3.37}$$

Ce problème est simple à résoudre :

$$\begin{cases} \text{Si } (h^k - T^k x^\nu)^T \pi^{kl} = \max_{l'} (h^k - T^k x^\nu)^T \pi^{kl'} : \alpha^l = 1 \\ \text{(si le max est atteint pour plusieurs } l\text{, on en choisit un seul)} \\ \text{Sinon, } \alpha^l = 0 \end{cases}$$

$$\begin{cases} \text{Si il existe } \sigma^{kl} \text{ tel que } (h^k - T^k x^\nu)^T \sigma^{kl} > 0\text{, le primal n'est pas réalisable} \\ \text{Sinon, } \forall l \in \{1,\dots,p\},\ \beta = 0 \end{cases}$$

Notons θ^k la valeur du problème $SD^{k\nu}$. Grâce au résultat précédent, nous pouvons

introduire deux familles de coupes au problème (PI) :

Réalisabilité : $(h^k - T^k x^\nu)^T \sigma^{kl} \leq 0,\ \forall l \in \{1, \ldots, q\}$

Optimalité : $\theta^k \geq (h^k - T^k x^\nu)^T \pi^{kl},\ \forall l \in \{1, \ldots, p\}$

Remarque 3.2.1. *La dualité forte nous permet alors d'affirmer que si une solution du dual vérifie toutes les coupes d'optimalité et de réalisabilité, alors :*

1. *Le problème primal est réalisable*
2. *La valeur optimale du primal est égale à celle du dual, c'est-à-dire* $\max_{l'}(h^k - T^k x^\nu)^T \pi^{kl'}$

Remarque 3.2.2. *Les points et directions extrêmes* σ^{kl} *et* π^{kl} *sont des constantes du problème, au sens où elles sont indépendantes de* x^ν*.*

On peut donc remplacer $(q^k)^T y^k$ par une variable θ^k, et ajouter à la fois les coupes d'optimalité et de réalisabilité. Si toutes ces coupes sont vérifiées, pour tout x la valeur optimale de θ^k associée sera effectivement $\mathcal{Q}(x)$. L problème (PI) s'écrit finalement :

$$
\begin{aligned}
(PLS)\ \min_{x,\theta} z &= c^T x + \sum_{k=1}^{K} p^k \theta^k && (3.38)\\
Ax &= b && (3.39)\\
(\sigma^{kl})^T T^k x &\geq (\sigma^{kl})^T h^k,\ \forall (k \in K, l \in \{1, \ldots, q\}) && (3.40)\\
\theta^k + (\pi^{kl})^T T^k x &\geq (\pi^{kl})^T h^k,\ \forall (k \in K, l \in \{1, \ldots, p\}) && (3.41)\\
x \geq 0 \quad &, \quad \theta \text{ libre} && (3.42)
\end{aligned}
$$

Le nombre de valeurs de k et l étant très élevés, il est trop coûteux d'énumérer toutes les coupes d'optimalité et de réalisabilité. De plus, chacune n'a pas pas forcément la même importance. La méthode *L-shaped* va les générer au fur et à mesure.

Remarque 3.2.3. *En fait, la méthode L-shaped consiste à minorer le plus finement possible* $\mathcal{Q}(x)$ *par des hyperplans.*

3.2.2 Algorithme

La méthode L-shaped est présentée dans l'algorithme 1.

Algorithme 1 Méthode L-shaped

$0 \leftarrow \nu$
Résoudre le problème maître, c'est-à-dire le problème $\min_{\{x|Ax=b,\ x\geq 0\}} c^T x$
while $x^{\nu+1} \neq x^\nu$ ou $\nu = 0$ **do**
  $\nu++$
  Tenter de résoudre $SP^k(x^\nu)$, pour tout $k \in \{1 \dots K\}$
  if $\exists k^* \mid SP^{k^*}(x^\nu)$ est non réalisable (cf. 3.2.3) **then**
    Trouver les $\sigma^l_{k^*}$
    Ajouter les coupes de réalisabilité associées
    Résoudre le problème maître augmenté (solution : $\bar{x}$)
    $x^\nu \leftarrow \bar{x}$
  else
    Chercher π^{kl} et ajouter une coupe d'optimalité par scénario pour borner θ
    Résoudre le problème maître augmenté (solution : $\bar{x}$)
    $x^\nu \leftarrow \bar{x}$
  end if
end while

L'algorithme ne boucle pas indéfiniment, puisqu'une fois que toutes les coupes ont été ajoutées (et elles sont en nombre fini), nous sommes en présence d'un problème équivalent au problème d'origine.

3.2.3 Génération des coupes de réalisabilité et d'optimalité

Si le problème $(SP)^{k\nu}$ est réalisable, `CPLEX` va, en le résolvant, renvoyer les variables duales associés à l'optimum primal : on obtient ainsi directement un point extrême de l'espace dual Λ^k, et par conséquent une coupe d'optimalité.

Remarque 3.2.4. *Lorsque le sous-problème est fortement dégénéré (*i.e. *il y a beaucoup de solutions duales optimales), on résout un sous-problème secondaire afin de générer des coupes dites* Pareto-optimales*, introduites par Magnanti et Wong (1981). Le but du sous-problème est de choisir la direction extrême ou le point extrême offrant la coupe la plus « sélective » parmi toutes celles disponibles. Récemment, Papadakos (2008) a amélioré l'algorithme de Magnanti et Wong, et a ainsi obtenu des gains significatifs de vitesse (facteur allant jusqu'à 7.50 sur ses instances).*

Afin de tester la réalisabilité de $(SP)^{k\nu}$ et de trouver une coupe de réalisabilité, on peut utiliser la méthode suivante.

Posons $e^T = (1, 1, \ldots, 1)$. Rappelons qu'à ce stade de l'algorithme, on dispose d'une solution x^ν. Définissons, pour tout $k \in \{1, \ldots, K\}$, le problème suivant :

$$\begin{aligned} \min_{y^k, v^+, v^-} w &= e^T v^+ + e^T v^- && (3.43) \\ W y^k + T^k x^\nu &= h^k - (v^+ - v^-) && (3.44) \\ y^k \geq 0, v^+ &\geq 0, v^- \geq 0 && (3.45) \end{aligned}$$

La valeur optimale w de ce programme est nulle si la contrainte $W y^k + T^k x^\nu = h^k$, sinon, elle est strictement positive. Nous allons montrer que, le cas échéant, le multiplicateur de Lagrange σ, obtenu en résolvant ce problème par la méthode du simplexe, fournit une direction extrême du cône Λ^k ne respectant pas les contraintes de réalisabilité. Si le problème est réalisable, on peut également montrer que la solution de ce problème est un point extrême de Λ^k ne respectant pas les contraintes d'optimalité.

Pour cela, associons à la contrainte d'égalité la variable duale σ, et aux contraintes de non-négativité de (y, v^+, v^-) les variables duales (α, β, γ) respectivement. Le problème dual s'écrit :

$$\begin{aligned} \max_{\sigma} w_D &= (h^k - T^k x^\nu)^T \sigma && (3.46) \\ W^T \sigma &\leq 0 && (3.47) \\ \sigma - e &\leq 0 && (3.48) \\ \sigma + e &\geq 0 && (3.49) \end{aligned}$$

Remarquons alors qu'un vecteur σ est une direction extrême du cône issu de la décomposition de $W^T \sigma \leq q^k$ si et seulement si il est un point / une direction extrême de $W^T \sigma \leq 0$. Puisque nous cherchons une solution optimale d'un problème linéaire, si σ est solution optimale du problème dual, il est un point / une direction extrême du polyèdre des contraintes, et donc une direction extrême de Λ^k.

De plus, lorsque les problèmes primal et dual sont réalisables, ils ont même valeur. Rechercher $w > 0$ est alors équivalent à rechercher $w_D > 0$, c'est-à-dire rechercher σ, direction extrême de Λ^k d'après le point précédent, tel que $(h^k - T^k x^\nu)^T \sigma > 0$, ce qui est exactement l'expression de la coupe de réalisabilité.

La résolution par la méthode du simplexe du problème auxiliaire permet donc d'obtenir directement les directions extrêmes du polyèdre Λ^k qui ne satisfont pas les contraintes de réalisabilité.

Remarque 3.2.5. *Les contraintes (3.48)-(3.49) sont uniquement des contraintes de normalisation, qui ne changent rien puisque l'on recherche une direction.*

Remarque 3.2.6. *La production de coupes de réalisabilité ne nous intéressera pas pour notre problème, car l'introduction d'une possibilité de sous-couverture rend le problème de recours toujours réalisable. De plus,* `CPLEX` *est également capable de renvoyer les directions extrêmes du polyèdre dual si le primal d'un problème est non réalisable : résoudre ou tenter de résoudre $(SP)^{k\nu}$ est donc suffisant en pratique.*

En pratique, la méthode L-shaped est efficace pour les problèmes linéaires stochastiques continus. Il en existe une variante pour les problèmes linéaires stochastiques en nombres entiers. Nous la présentons, mais elle est peu utilisée.

3.3 La méthode L-shaped en nombres entiers

3.3.1 Cadre général

Il est possible de généraliser la méthode L-shaped pour les problèmes stochastiques en nombres entiers. Nous reprenons encore une fois dans cette section le plan et les notations du livre de Birge et Louveaux (1997).

Considérons le problème suivant (que nous nommerons *problème courant*) :

$$
\begin{aligned}
(PC) \min_{x \in X} z &= c^T x + \theta && (3.50)\\
Ax &= b && (3.51)\\
D_l x &\geq d_l,\ l \in \{1, \ldots, r\} && (3.52)
\end{aligned}
$$

$$E_l x + \theta \geq e_l,\ l \in \{1, \ldots, s\} \tag{3.53}$$

$$x \in X \quad , \quad \theta \in \mathbb{R} \tag{3.54}$$

Les coupes (3.52) et (3.53) représentent respectivement des coupes (quelconques) de réalisabilité et d'optimalité. Ce sont des généralisations des contraintes (3.40) et (3.41). Ce problème courant correspondrait en fait à un problème intermédiaire issu de la méthode L-shaped, si nous étions dans le cas continu. On a $X = \bar{X} \cap \mathbb{R}^n_+$ ou $X = \bar{X} \cap \mathbb{Z}^n_+$ suivant les cas, et $\bar{X}$ représente un ensemble de contraintes que l'on souhaitera relâcher (par exemple, $T^k x + Wy = h^k$) comme lors de la méthode L-shaped continue.

Remarque 3.3.1. *Notons que (PC), pour (r, s) quelconque, n'est pas équivalent au problème initial (PLS) en nombres entiers. Il n'en possède qu'une partie des contraintes de réalisabilité et d'optimalité.*

Afin d'introduire un cadre général pour la méthode L-shaped pour les problèmes en nombres entiers, donnons deux définitions.

Définition 3.3.1. *Un jeu de coupes de réalisabilité est dit* valide *pour $x \in X$ s'il existe un entier r tel que :*

$$x \in \{x | D_l x \geq d_l,\ l = 1,\ \ldots,\ r\} \Rightarrow x \in \bar{X}$$

Définition 3.3.2. *Un jeu de coupes d'optimalité est dit* valide *pour $x \in X$ s'il existe un entier s tel que :*

$$(x, \theta) \in \{(x, \theta) | E_l x + \theta \geq e_l, l = 1,\ \ldots,\ s\} \Rightarrow \theta \geq \mathcal{Q}(x)$$

Remarquons que ces deux propriétés sont à la base de la méthode de Benders. Les contraintes de réalisabilité (3.40) et d'optimalité (3.41) les satisfont dans le problème (PLS), c'est précisément ce qui nous a permis d'appliquer la méthode L-shaped. Dans ce cas, l'ensemble X était $\{x \in \mathbb{R}^{n_1} | \ \exists y \ / \ T^k x + Wy = h^k, k = 1,\ \ldots,\ K\}$. Supposons que l'on dispose de telles coupes pour nos problèmes entiers, il serait possible d'utiliser une méthode L-shaped.

Proposition 3.3.1. *Si l'on suppose que :*

– *Pour tout $x \in X$, $\mathcal{Q}(x)$ est calculable en un nombre fini d'étapes*

- *On dispose d'un jeu de coupes de réalisabilité valide pour notre problème*
- *On dispose d'un jeu de coupes d'optimalité valide pour notre problème*

Alors la méthode L-shaped pour les programmes stochastiques en nombres entiers fournit une solution optimale (si elle existe) en un nombre fini d'étapes.

La démonstration de cette proposition repose sur le fait qu'à partir d'une certaine étape, toutes les coupes d'optimalité et de réalisabilité du problème auront été ajoutées, nous donnant alors un problème équivalent au problème initial. Bien évidemment, le but est de pouvoir s'arrêter bien avant.

Dans le cas où les contraintes d'intégrité ne portent que sur les variables primaires de décision x, il est toujours possible d'utiliser la théorie de la dualité pour le problème secondaire, et par conséquent d'obtenir les mêmes coupes que précédemment. La méthode L-shaped pour les nombres entiers *peut donc s'appliquer sans autre modification* dans ce cas[2]. Dans le cas où les contraintes d'intégrité portent également sur le problème secondaire, on utilise les coupes décrites ci-après.

3.3.2 Coupes de réalisabilité

Notons :

$$K_2(\xi) = \{x \mid \exists y,\ Wy = h(\omega) - T(\omega)x, y \in \mathbb{Z}_+^{n_2}\} \tag{3.55}$$

$$C_2(\xi) = \{x \mid \exists y,\ Wy = h(\omega) - T(\omega)x, y \in \mathbb{R}_+^{n_2}\} \tag{3.56}$$

On note $(P1)$ le problème :

$$w^k(x) = \min e^T v^+ + e^T v^- \tag{3.57}$$

$$Wy + v^+ + v^- = h^k - T^k x \tag{3.58}$$

$$y \in \mathbb{Z}_+^{n_2}, v^+ \geq 0, v^- \geq 0 \tag{3.59}$$

On résout le problème relâché continu de $(P1)$. Deux cas se produisent alors :

[2]On l'utilise alors conjointement avec un algorithme de branch and bound.

Si la solution y est entière : si $w > 0$, $x \notin C_2(\xi)$. On peut alors appliquer le même type de coupe que dans le cas continu. Sinon, cela signifie que le problème (SP^k) en nombres entiers est réalisable.

Si la solution y n'est pas entière : On branche sur les valeurs de y et on recommence ($y_i \leq 0$ et $y_i \geq 1$ par exemple).

On indexe par $\rho \in \{1, \ldots, R\}$ les nœuds terminaux de l'arbre. On note Y^ρ les régions associées (qui forment une partition de $\mathbb{R}^{n_2}_+$). On a :

$$x \in K_2(\xi^k) \quad \Leftrightarrow \quad x \in \cup_{p=1}^{R} K_2^\rho(\xi^k) \tag{3.60}$$

$$\text{avec : } K_2^\rho(\xi^k) \quad = \quad \{x \mid \exists y \in Y^\rho \text{ tel que } Wy = h^k - T^k x,\ y \geq 0\} \tag{3.61}$$

On peut montrer que $K_2^\rho(\xi^k)$ est un polyèdre. Puisque, pour que le problème (SP^k) en nombres entiers soit réalisable, il faut (par définition), que $x \in K_2(\xi^k)$, cela signifie qu'il faut que x soit dans au moins un des $K_2^\rho(\xi_K)$. Ces espaces étant des polyèdres, il est possible de les caractériser entièrement par des coupes linéaires. Ainsi, un jeu valide de coupes de réalisabilité sera constitué d'une contrainte demandant à ce qu'au moins un des R ensembles de coupes linéaires ainsi obtenus soit satisfait.

En pratique, on résout le problème pour $\bar{x}$ et on choisit une coupe par valeur de ρ (donc par région $K_2^\rho(\xi^k)$), et on demande à ce qu'au moins une de ces R coupes soit satisfaite. Cela se fait par l'ajout de variables binaires δ_ρ au problème. En supposant que la coupe d'intégrité choisie sur Y^ρ s'écrive $u_\rho^T x \leq d_\rho$, la coupe globale s'écrit :

$$u_\rho^T x \quad \leq \quad d_\rho + M_\rho(1 - \delta_\rho),\ \rho \in \{1, \ldots, R\} \tag{3.62}$$

$$\sum_{\rho=1}^{R} \delta_\rho \quad \geq 1 \tag{3.63}$$

$$\delta \quad \in \quad \{0,1\}^R \tag{3.64}$$

3.3.3 Coupes d'optimalité

On cherche une coupe d'optimalité sur θ^k. En utilisant les nœuds terminaux de l'arbre de branchement décrit dans la section précédente, on obtient une partition de $\mathbb{R}_+^{n_2}$ de la forme :

$$\begin{aligned} \mathbb{R}_+^{n_2} &= \cup_{\rho=1}^{R} Y^\rho && (3.65)\\ Y^\rho &= \{y \mid a^\rho \leq y \leq b^\rho\} && (3.66)\\ Y^{\rho_1} \cap Y^{\rho_2} &= \emptyset,\ \forall \rho_1 \neq \rho_2 && (3.67) \end{aligned}$$

La fonction de recours sur une feuille Y^ρ s'écrit donc :

$$Q^\rho(x^\nu, \xi^k) = \min\{q^T y \mid Wy = h^k - T^k x^\nu,\ a^\rho \leq y \leq b^\rho\} \quad (3.68)$$

En écrivant le problème dual, et en associant π à la contrainte d'égalité, α à celle de supériorité et β à celle d'infériorité, il vient à l'optimum (sous réserve de réalisabilité) :

$$Q^\rho(x^\nu, \xi^k) = (h^k - T^k x^\nu)^T \pi^\rho + (a^\rho)^T \alpha^\rho + (b^\rho)^T (\beta^\rho) \quad (3.69)$$

En posant :

$$\begin{aligned} (\sigma_\rho^k)^T &:= (-\pi^\rho)^T T^k && (3.70)\\ \tau_\rho^k &:= (\pi^\rho)^T h^k + (\alpha^\rho)^T a^\rho + (\beta^\rho)^T b^\rho && (3.71)\\ &\Rightarrow\ Q^\rho(x^\nu, \xi^k) = (\sigma_\rho^k)^T x^\nu + \tau_\rho^k && (3.72) \end{aligned}$$

Soit x la solution optimale du problème. On a, par dualité[3] :

$$Q^\rho(x, \xi^k) \geq (\sigma_\rho^k)^T x^\nu + \tau_\rho^k \quad (3.73)$$

[3]Le dual étant un problème de *maximisation*

De plus :

$$Q(x, \xi^k) = \min_{\rho \in \{1,\dots,R\}} Q^\rho(x, \xi^k) \tag{3.74}$$

Par conséquent :

$$\theta^k \geq \left(\min_{\rho \in \{1 \dots R\}} (\sigma_\rho^k)^T x^\nu + \tau_\rho^k \right) \tag{3.75}$$

où θ^k est la valeur optimale du problème $(SP)^{k\nu}$. On a $\theta = \sum_k p^k \theta^k$. Si certains nœuds terminaux ne sont pas réalisables, on les oublie.

Conclusion de la partie

On a présenté ici la classe des problèmes linéaires stochastiques en deux étapes, classe à laquelle appartient le problème que l'on cherche à résoudre. Nous avons également présenté deux méthodes de résolution, inspirées de la décomposition de Benders (1962). La méthode L-shaped pour les problèmes continus est efficace et utilisée.

En revanche, son adaptation pour les problèmes en nombres entiers est lourde à mettre en place et peu utilisée en pratique. Nous allons donc présenter dans la partie suivante d'autres méthodes permettant de résoudre des problèmes linéaires stochastiques en nombres entiers. Nous verrons qu'aucune d'entre elles ne semble non plus être adaptée à notre problème.

Chapitre 4

ALGORITHMES DE RÉSOLUTION DES PSNE

Dans notre cas, il est impossible d'employer directement la méthode L-shaped, car nous avons en plus des contraintes d'intégrité sur nos variables secondaires. La méthode L-shaped pour les problèmes en nombres entiers étant peu efficace, d'autres procédures ont été développées afin de résoudre de tels problèmes. Un grand nombre d'entre elles est présenté dans le chapitre de Sherali et Zhu (2009).

Toutes sont basées sur une décomposition du problème en problème maître et sous-problèmes. La recherche dans ce domaine est toujours active, et aucune méthode ne fait consensus. Comme dans beaucoup de problèmes complexes de recherche opérationnelle, il semble difficile de trouver une méthode « universelle » résolvant tous les problèmes stochastiques en nombres entiers en un temps raisonnable. Le mieux à faire alors est de sélectionner et d'adapter, pour un problème donné, la méthode qui semble la plus apte à le résoudre.

En raison de leur mauvaise adaptation à notre type de problème (peu de scénarios mais beaucoup de variables et de contraintes), nous n'implémenterons aucune de ces méthodes, et développerons notre propre heuristique. Nous les présentons néanmoins afin d'avoir un bref état de l'art de la résolution exacte des PSNE.

4.1 Relaxation lagrangienne

Nous présentons ici une méthode présentée par Caroe et Schultz (1999). Reprenons les notations précédentes et définissons les ensembles $S^k,\ k \in \{1, \dots K, \}$:

$$S^k = \left\{(x, y^k) \mid Ax = b,\ x \in X,\ T^k x + W y^k = h^k,\ y^k \in Y\right\} \tag{4.1}$$

Les ensembles $X \subseteq \mathbb{R}_+^{n_1}$ et $Y \subseteq \mathbb{R}_+^{n_2}$ représentent les contraintes d'intégrité ou de binarité sur x et y. Le problème déterministe équivalent peut donc s'écrire :

$$z = \min \left\{ c^T x + \sum_{k=1}^{K} p^k (q^k)^T y^k \mid (x, y^k) \in S^k,\ k = 1, \ldots, K \right\} \tag{4.2}$$

On suppose que ce problème est réalisable et borné. L'idée de la méthode est de décomposer ce problème en ses différents scénarios, en « copiant » K fois la variable x, puis en imposant l'égalité des K variables ainsi crées. Le problème se reformule alors ainsi :

$$\min \left\{ \sum_{k=1}^{K} p^k (c^T x^k + (q^k)^T y^k) \mid (x^k, y^k) \in S^k, k = 1, \ldots,\ K, x^1 = \cdots = x^K \right\} \tag{4.3}$$

La contrainte couplante est appelée contrainte de *non-anticipativité*. L'idée va être de relaxer cette contrainte. Pour cela, nous allons noter $H = (H_1, \ldots, H_K)$ une matrice telle que la contrainte $x_1 = \cdots = x_K$ s'écrive sous la forme :

$$\sum_{k=1}^{K} H^k x^k = 0 \tag{4.4}$$

Posons alors :

$$L(\lambda) \quad = \quad \min \left\{ \sum_{k=1}^{K} \left(p^k (c^T x^k + (q^k)^T y^k) + \lambda (H^k x^k) \right) \mid \forall k \in \{1, \ldots K\},\ (x^k, y^k) \in S^k \right\}$$

On peut ainsi écrire le problème dual de (4.3) :

$$z_{LD} = \max_{\lambda} L(\lambda) \tag{4.5}$$

Par dualité faible, z_{LD} est une borne inférieure pour le problème (4.3). De plus, si pour un choix quelconque de λ, la solution $(x^k, y^k),\ k = 1, \ldots, K$ issue du problème

$L(\lambda)$ est compatible avec le problème (4.3), alors cette solution est optimale pour le problème de départ, et λ est solution du problème (4.5).

Remarque 4.1.1. *Puisque nous travaillons avec des contraintes d'intégrité, la dualité forte ne peut s'appliquer, et l'on n'a donc pas forcément égalité entre z_{LD} et la valeur optimale du problème de départ.*

Le problème a été posé de manière à ce que la fonction lagrangienne puisse se décomposer facilement, de sorte que :

$$L(\lambda) = \sum_{k=1}^{K} L^k(\lambda) \tag{4.6}$$

$$L^k(\lambda) = \min\left\{(p^k(c^T x^k + (q^k)^T y^k) + \lambda(H^k x^k) \mid (x^k, y^k) \in S^k\right\} \tag{4.7}$$

Cela permet de calculer plus facilement z_{LD}, grâce à une méthode de *sous-gradient*. Il s'ensuit alors un algorithme de branch and bound.

Remarque 4.1.2. *Il est également possible de diminuer, via un jeu d'écriture, la taille du vecteur des multiplicateurs de Lagrange λ lorsque certaines variables doivent être non seulement entières, mais binaires.*

4.2 Partitionnement de la fonction valeur

Nous présentons ici une méthode proposée par Ahmed *et al.* (2004). Il a été montré que la fonction valeur des problèmes stochastiques en nombres entiers est seulement semi-continue inférieurement par rapport aux variables de décision primaires. Ahmed *et al.* expliquent alors que, en raison de ce cadre semi-continu, les algorithmes de branch and bound déjà développés pour résoudre de tels problèmes avec une ou plusieurs variables primaires continues sont certes convergents, mais convergent en général en temps infini (y compris la méthode développée par Schultz *et al.* (1998), servant dans leur article de point de comparaison).

Ahmed *et al.* montrent que, dans le cas où la région réalisable X du problème maître est compacte (on travaille donc en continu sur toutes variables primaires de décision, quitte à enchaîner avec un algorithme de branch and bound dans le cas où certaines variables primaires seraient entières), ces discontinuités sont en nombre fini.

Il note par ailleurs qu'elles ne sont pas forcément orthogonales aux variables de décision, ce qui empêche un algorithme de branch and bound agissant sur les variables de décision de converger en temps fini. En effet, en branchant directement sur les variables de décision, on obtient des hyper-rectangles de l'espace de départ. En observant la figure 4.1, on constate qu'il ne sera pas dans tous les cas possibles de faire converger en un temps fini borne inférieure et supérieure d'un tel hyper-rectangle, lorsqu'il comporte une discontinuité, en branchant seulement sur les valeurs de ces variables.

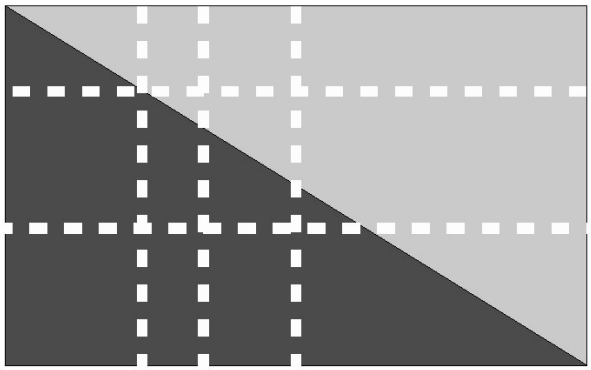


FIGURE 4.1 Problème de discontinuité

Ahmed *et al.* proposent une reformulation du problème ainsi qu'un algorithme de branch and bound permettant de trouver une solution optimale en un temps fini, même en présence de variables primaires continues.

4.2.1 Principe

Rappelons d'abord la formulation du problème d'origine :

$$(PI) \quad \min \quad c^T x + \sum_{k=1}^{K} p^k Q^k(x) \tag{4.8}$$

$$x \in X \tag{4.9}$$

avec :

$$Q^k(x) = \min \quad q^T y \tag{4.10}$$
$$\text{s.l.c.} \quad W^k y \geq h^k + Tx \tag{4.11}$$
$$y \in Y \cap \mathbb{Z}^{n_2} \tag{4.12}$$

Remarque 4.2.1. *Contrairement aux cas précédents, ici W peut dépendre de l'aléa mais T est fixée. Dans notre problème, à la fois W et T sont fixés, puisque seule la demande (équivalente de h^k) varie.*

Remarque 4.2.2. *On travaille ici avec une contrainte de non-anticipativité (4.11) qui n'est plus une égalité mais une inégalité.*

On fait les hypothèses suivantes (m_2 est le nombre de contraintes du problème secondaire) :

- X est non-vide et compact.
- Pour tout $x \in X$ et pour tout k, $Q^k(x) < \infty$.
- Pour tout k, la matrice W^k est entière.
- Pour tout k, il existe $u^k \in \mathbb{R}_+^{m_2}$ tel que $u^k W^k \leq q^k$.

La dernière hypothèse permet de garantir que $Q^k(x) > -\infty$. Le problème est alors reformulé comme ceci :

$$(TP) \quad \min \quad f(\chi) \tag{4.13}$$
$$\text{s.l.c.} \quad \chi \in \mathcal{X} \tag{4.14}$$

Avec :

$$f(\chi) = \phi(\chi) + \bar{\psi}(\chi) \tag{4.15}$$
$$\bar{\psi}(\chi) = \sum_{k=1}^{K} p^k \psi^k(\chi) \tag{4.16}$$

$$\phi(\chi) = \min\{c^T x | Tx = \chi,\ x \in X\} \tag{4.17}$$

$$\psi^k(\chi) = \min\{(q^k)^T y | W^k y \geq h^k + \chi,\ y \in Y \cap \mathbb{Z}^{n_2}\} \tag{4.18}$$

$$\mathcal{X} = \{\chi \in \mathbb{R}^{m_2} | \chi = Tx,\ x \in X\} \tag{4.19}$$

Les variables χ sont parfois appelées *tender variables*. Ahmed *et al.* (2004) montrent que cette reformulation est bien équivalente au probleme d'origine.

Théorème 4.2.1. *Soit χ^* une solution de (TP). Alors $x^* \in argmin\{c^T x | x \in X,\ Tx = \chi^*\}$ est une solution optimale de (PI). De plus les valeurs optimales de ces deux problèmes sont égales.*

Nous allons par la suite esquisser la preuve établie par Ahmed *et al.* montrant que, sur certaines régions, $\bar{\psi}(\chi)$ est constante. L'idée sera alors de brancher sur ces régions, qui formeront qui plus est une partition de $\mathbb{R}^{m_2}$. Notons $\psi_j^k(\chi_j)$ la partie de la fonction $\psi^k(\chi)$ due à la composante $j \in \{1 \dots m_2\}$ de χ.

Lemme 4.2.1. *Pour tout $k = 1, \dots, K$ et pour tout $j = 1, \dots, m_2$, $\psi_j^k(\chi_j)$ est semi-continue inférieure croissante en χ_j.*

Lemme 4.2.2. *Pour tout $z \in \mathbb{Z}$, $\psi_j^k(\chi_j)$ est constante sur l'intervalle $\chi_j \in]z - h_j^k - 1,\ z - h_j^k]$, pour tout $k = 1, \dots, K$ et $j = 1, \dots, m_2$.*

Démonstration. On a supposé que W était entière. Par conséquent, puisque y est entière également, $(W^k y)_j \geq h_j^k + \chi_j \Rightarrow (W^k y)_j \geq \lceil h_j^k + \chi_j \rceil$. Donc, pour tout $z \in \mathbb{Z}$, $\psi_j^k(\chi_j)$ est constante sur l'ensemble $\{(h_j^k + \chi_j) - 1 < h_j^k + \chi_j \leq k\}$. De manière équivalente, $\psi_j^k(\chi_j)$ est constante sur l'intervalle $\chi_j \in]z - h_j^k - 1, z - h_j^k[$, pour tout $z \in \mathbb{Z}$. □

Théorème 4.2.2. *Soit $\mathbf{z} = (z_1^1, \dots, z_j^k, \dots, z_K^{m_2})^T \in \mathbb{Z}^{Km_2}$. Pour $\mathbf{z}$ fixé, posons :*

$$\mathcal{C}(\mathbf{z}) := \left\{ \chi \in \mathbb{R}^{m_2} | \chi \in \cap_{k=1}^{K} \prod_{j=1}^{m_2}]z_j^k - h_j^k - 1,\ z_j^k - h_j^k] \right\} \tag{4.20}$$

On a alors :

1. *Si $\mathcal{C}(\mathbf{z}) \neq \emptyset$, alors sa fermeture est un hyper-rectangle de dimension maximale (i.e. m_2).*

2. *L'ensemble* $\{\mathcal{C}(\mathbf{z})|\mathbf{z} \in \mathbb{Z}^{Km_2}\}$ *forme une partition de* $\mathbb{R}^{m_2}$.
3. *Si* $\mathcal{C}(\mathbf{z}) \neq \emptyset$, *alors* $\bar{\psi}(\chi)$ *est* constante *sur* $\mathcal{C}(\mathbf{z})$.

Ainsi, la reformulation a permis de construire des hyper-rectangles de $\mathbb{R}^{m_2}$ sur lesquels la fonction valeur, en plus d'être continue, est constante. Un autre théorème affirme que, si $\mathcal{X}$ est compact, alors ces hyper-rectangles sont en nombre fini.

Théorème 4.2.3. *Soit* $\mathcal{X} \subset \mathbb{R}^{m_2}$ *et* $\mathcal{Z} := \{z \in \mathbb{Z}^{Km_2}|\mathcal{C}(\mathbf{z}) \cap \mathcal{X} \neq \emptyset\}$. *Si* $\mathcal{X}$ *est compact, alors* $|\mathcal{Z}| < \infty$.

4.2.2 Application

Ahmed *et al.* utilisent le cadre classique des algorithmes de branch and bound, en utilisant le partitionnement de $\mathcal{X}$ créé à la section précédente. Nous présentons ici les principales subtilités de l'algorithme.

Calcul de départ

Avant de commencer l'algorithme, Ahmed *et al.* construisent un hyper-rectangle de départ $\mathcal{P}^0$, tel que $\mathcal{X} \subset \mathcal{P}^0$. Pour cela, il regarde pour chaque composante j sa valeur minimale possible l_j et sa valeur maximale possible u_j, indépendamment des autres composantes (rappelons que l'on a supposé $\mathcal{X}$ compact). En encadrant chaque χ_j par l_j et u_j, il forme ainsi un polyèdre contenant $\mathcal{X}$. Il cherche ensuite le plus petit hyper-rectangle contenant ce polyèdre.

Borne inférieure

On résout le problème suivant :

$$
\begin{aligned}
f_L(\mathcal{P}^k) = \quad & \min \quad c^T x + \theta && (4.21)\\
& \text{slc} \quad x \in X,\ Tx = \chi && (4.22)\\
& \qquad l^k \leq \chi \leq u^k && (4.23)\\
& \qquad \theta \geq \sum_{k=1}^{K} p^k \psi^k (l^k + \epsilon) && (4.24)
\end{aligned}
$$

où l'on a :

$$\begin{aligned} \psi^k(\chi) = \quad & \min \quad (q^k)^T y && (4.25)\\ & \text{slc} \quad W^k y \leq h^k + \chi && (4.26)\\ & \qquad y \in Y \cap \mathbb{Z}^{n_2} && (4.27) \end{aligned}$$

et où l'on aura choisi ϵ tel que $\psi^k(\cdot)$ soit constante sur $]l^k, l^k+\epsilon]$. On procède donc en deux étapes. On résout d'abord le problème (4.25)-(4.27). Il est montré dans l'article que ψ est croissante. Le sous-problème nous donne donc une borne inférieure de ψ sur l'hyper-rectangle $\mathcal{P}^k$. On résout ensuite le problème (4.21)-(4.24), qui cherche lui une borne inférieure de $c^T x$ sur l'hyper-rectangle. On obtient finalement un minorant du problème sur $\mathcal{P}^k$, où coût du problème primaire et coût du problème secondaire sont découplés.

Remarque 4.2.3. *Suivant les cas, les deux problèmes précédents peuvent être en nombres entiers, ce qui rallonge considérablement leur résolution.*

Borne supérieure

On se contente de trouver une solution réalisable, qui sera ainsi une borne supérieure sur la valeur optimale du problème de minimisation.

Branchement

Une solution simple serait de partitionner l'hyper-rectangle en le coupant en deux sur son côté le plus long. Ce schéma de branchement, bien que *complet*, c'est-à-dire menant à un partitionnement complet de l'ensemble de départ, ne permet pas de se débarrasser des discontinuités de la fonction valeur. Les auteurs ont donc retenu de couper l'hyper-rectangle courant suivant un axe j' là où il y a le plus de chances de se produire une discontinuité, et présente une méthode pour y parvenir.

4.2.3 Discussion et résultats

Ahmed compare sa méthode avec celle de Caroe et Schultz (1999), basée sur une relaxation lagrangienne. Sur les instances testées, de petite taille, sa méthode est beaucoup plus rapide (facteur d'ordre 10). Pour les problèmes plus grands (variables

binaires, continues et nombre de contraintes de l'ordre de 100), l'algorithme de Carøe et Schultz ne termine pas. Celui d'Ahmed *et al.* termine plus souvent (deux instances sur trois). Néanmoins, pour le plus gros problème (10 scénarios), aucun des deux algorithmes n'est meilleur (aucun ne termine, et les gaps entre borne supérieure et borne inférieure sont quasiment identiques). La méthode d'Ahmed *et al.* se rélève également plus efficace que celle de Schultz *et al.* (1998).

Dans un autre article, Ahmed et Garcia (2004) appliquent cet algorithme à un problème de plus grande échelle avec succès, là où `CPLEX` n'y parvient pas. A la vue des résultats, il propose deux améliorations possibles. Tout d'abord, en remarquant que `CPLEX` converge assez vite vers une solution de bonne qualité (mais pas optimale), il suggère d'utiliser `CPLEX` afin de calculer une « bonne » solution de départ pour son algorithme de branch and bound. Il constate également que la résolution des problèmes de bornes inférieures est un des goulots d'étranglement de leur algorithme, et suggère d'améliorer la méthode permettant de les déterminer.

Outre sa rapidité, le principal intérêt de l'algorithme proposé par Ahmed réside dans le fait que, même lorsque les variables primaires sont continues, l'algorithme converge en temps fini vers une solution optimale. Dans notre cas, ce « luxe » est-il nécessaire ? En effet, toutes les variables de notre problème maître (les variables S_j) sont entières.

4.3 Superadditivité duale

Dans leur article, Kong *et al.* (2006) proposent une méthode de résolution des problèmes stochastiques en nombres tous entiers. Le principe de cette méthode est de calculer explicitement, grâce à des résultats issus de la théorie de la dualité pour les problèmes en nombres entiers, toutes les valeurs des fonctions valeur ψ et ϕ définies dans la section 4.2. S'ensuit alors une exploration de toutes ces valeurs afin d'en déterminer la solution optimale.

4.3.1 Cadre de travail

Nous reprenons ici le problème présenté dans Kong *et al.* (2006), qui est un problème de maximisation :

$$\begin{aligned} \max \quad & c^T x + \mathbb{E}_\xi Q(x, \xi(\omega)) && (4.28)\\ \text{s.l.c.} \quad & Ax \leq b && (4.29)\\ & x \in \mathbb{Z}_+^{n_1} && (4.30) \end{aligned}$$

avec :

$$\begin{aligned} Q(x, \xi(\omega)) &= \max q^T y && (4.31)\\ \text{s.l.c. } Wy &\leq h(\omega) - Tx && (4.32)\\ y &\in \mathbb{Z}_+^{n_2} && (4.33) \end{aligned}$$

En plus des hypothèses effectuées dans la section 4.2, on suppose que :
- q est fixé
- $x \in \mathbb{Z}_+^{n_1}$
- A, T, b et $h(\omega)$ sont entiers

4.3.2 Dualité pour les problèmes en nombres entiers

La dualité pour les problèmes en nombres entiers est une adaptation de la théorie de la dualité pour les problèmes linéaires. Nous présentons brièvement les résultats qui nous intéresseront.

Etant donnée une matrice $G \in \mathbb{Z}^{m \times n}$, on considère la famille paramétrée par $\beta \in \mathbb{Z}^m$ de problèmes en nombres entiers suivante :

$$(PIP) : \; z(\beta) = \max \left\{ \gamma^T x | x \in S(\beta) \right\}, \; S(\beta) = \left\{ x \in \mathbb{Z}_+^n | Gx \leq \beta \right\} \qquad (4.34)$$

La fonction $z(\cdot) : \mathbb{Z}^m \to \mathbb{Z}$ est appelée *fonction valeur* de (PIP). On la définit comme étant égale à $-\infty$ ou $+\infty$ si, respectivement, $S(\beta) = \emptyset$ ou si elle n'est pas bornée. On désigne par z_{LP} et S_{LP} les relaxations continues respectives de z et S. Etudions maintenant quelques propriétés de cette famille paramétrée.

Proposition 4.3.1. *$z(0) \in \{0,\ \infty\}$. Si $z(0) = 0$, alors $z(\beta) \leq \infty$ pour tout $\beta \in \mathbb{Z}^m$. Si $z(0) = \infty$, alors $z(\beta) = \pm\infty$ pour tout $\beta \in \mathbb{Z}^m$.*

Démonstration. Si un vecteur x_0 est dans $S(0)$, alors tout multiple de ce vecteur y est également. On en déduit immédiatement que $z(0) = +\infty$ s'il existe un vecteur $x_0 \in S(0)$ tel que $\gamma^T x_0 > 0$, 0 sinon. Le reste en découle. □

Proposition 4.3.2. *Soit g_j et γ_j respectivement la j-ième colonne de la matrice des contraintes G et le j-ième coefficient de la fonction objectif de (PIP). Alors $z(g_j) \geq \gamma_j$, pour $j = 1 \ldots n$.*

Démonstration. On a $S(g_j) = \{x | Gx \leq g_j\}$. En notant e_j le j-ième vecteur de base, on a $Ge_j = g_j$, par conséquent $e_j \in S(g_j)$. De plus, $\gamma^T e_j = \gamma_j$. On obtient donc une borne inférieure pour $z(g_j)$. □

Proposition 4.3.3 (Croissance)**.** *La fonction valeur de (PIP) est croissante sur $\mathbb{Z}^m$.*

Proposition 4.3.4 (Superadditivité)**.** *La fonction valeur de (PIP) est superadditive sur $D = \{\beta \in \mathbb{Z}^m | S(\beta) \neq \emptyset\}$, c'est-à-dire que pour tout $\beta_1,\ \beta_2 \in D$, si $\beta_1 + \beta_2 \in D$, on a :*

$$z(\beta_1) + z(\beta_2) \leq z(\beta_1 + \beta_2) \tag{4.35}$$

Proposition 4.3.5 (Condition de complémentarité entière)**.** *Si $\hat{x} \in opt(\beta)$, alors, $\forall x \in \mathbb{Z}^n_+$ tel que $x \leq \hat{x}$, on a la* condition de complémentarité *suivante :*

$$z(Gx) = \gamma^T x \text{ et } z(Gx) + z(\beta - Gx) = z(Gx) + z(G(\hat{x} - x)) = z(\beta) \tag{4.36}$$

Corollaire 4.3.1 (Elimination de colonnes)**.** *Si $z(g_j) > \gamma_j$, alors $\forall \beta \in \mathbb{Z}^m$ et $\forall \hat{x} \in opt(\beta)$, $\hat{x}_j = 0$.*

4.3.3 Reformulation et évaluation de la fonction duale

Comme Ahmed *et al.* (2004) (section 4.2), Kong *et al.* utilisent la formulation en *tender variables* afin de faire apparaître des fonctions valeurs qu'il va ensuite calculer de manière exhaustive.

$$(P2) : \max_{\beta \in \mathbf{B}^1} \phi(\beta) + \mathbb{E}_\xi \psi(h(\omega) - \beta) \tag{4.37}$$

avec :

$$\begin{aligned}
\phi(\beta_1) &= \max\{c^T x | x \in S_1(\beta_1)\} && (4.38)\\
S_1(\beta_1) &= \{x \in \mathbb{Z}_+^{n_1} | Ax \leq b,\ Tx \leq \beta_1\} && (4.39)\\
\psi(\beta_2) &= \max\{q^T y | y \in S_2(\beta_2)\} && (4.40)\\
S_2(\beta_2) &= \{x \in \mathbb{Z}_+^{n_1} | Wy \leq \beta_2\} && (4.41)\\
\mathbf{B}^1 &= \{\beta_1 \in \mathbb{R}^{m_2} | \exists x \in X, \beta_1 = Tx\} && (4.42)\\
\mathbf{B}^2 &= \cup_{\beta_1 \in \mathbf{B}^1} \cup_{\omega \in \Omega} \{h(\omega) - \beta_1\} && (4.43)
\end{aligned}$$

Comme dans Ahmed *et al.* (2004), on peut montrer que ce problème est équivalent à celui de départ (avec une inégalité comme contrainte de non-anticipativité).

Théorème 4.3.1. *Soit β^* une solution optimale de $(P2)$. Alors $\hat{x} \in$ argmax $\{c^T x | Ax \leq b,\ Tx \leq \beta^*,\ x \in \mathbb{Z}_+^{n_1}\}$ est une solution optimale du problème stochastique en nombres entiers de départ. De plus, ces problèmes ont même valeur.*

L'idée est alors d'évaluer point à point sur tout $\mathbf{B}^1$ et $\mathbf{B}^2$ les deux fonctions ϕ et ψ. Même si l'opération peut paraître longue et fastidieuse, elle est accélérée par les propriétés de dualité étudiées précédemment. Dans l'article de Kong *et al.* (2006), deux méthodes sont présentées pour effectuer ce calcul. Une basée sur la programmation en nombres entiers et une autre basée sur la programmation dynamique. Cette dernière demandant en plus que G soit positive et étant surtout efficace pour les petites instances, nous présenterons seulement la première.

On note $z(\cdot) : \beta \mapsto z(\beta)$ la fonction valeur d'un problème paramétré en nombres entiers de la forme de (PIP). L'algorithme consiste à encadrer, à chaque itération k et pour tout $\beta \in \mathbf{B}$, $z(\beta)$ par une borne inférieure $l^k(\beta)$ et par une borne supérieure $u^k(\beta)$. Lorsque $l^k(\beta) = u^k(\beta)$, on en déduit la valeur de $z(\beta)$.

Théorème 4.3.2. *L'algorithme 2 calcule en un temps fini toutes les valeurs de $z(\cdot)$.*

Algorithme 2 Evaluation de la fonction valeur

Etape 0 : Initialiser la borne inférieure $l^0(\beta) = -\infty$ pour tout $\beta \in \mathbf{B}$. Pour $j = 1, \cdots, n$, si $g_j \in \mathbf{B}$, poser $l^0(g_j) = \gamma_j$ (propriété 4.3.2). Résoudre z_{LP} pour un $\beta_0 \in \mathbf{B}$ arbitraire afin d'obtenir une solution duale optimale π_0. Initialiser alors, pour tout $\beta \in \mathbf{B}$, la borne supérieure $u^0(\beta)$ en posant : $u^0(\beta) = \pi_0^T \beta$. En effet, en écrivant le dual, on remarque que sa valeur est $\pi\beta$ et que la seule contrainte est $\gamma^T - \pi^T G = 0$. Par conséquent, pour tout β, π_0 reste réalisable pour le dual. La dualité faible permet ensuite d'affirmer que $\pi_0\beta \geq z_{LP}(\beta) \geq z(\beta)$. Passer à l'étape suivante, $k++$.

Etape 1 : Sélectionner une valeur de β pour laquelle $l^k(\beta) < u^k(\beta)$. Résoudre le problème en nombre entiers associé pour trouver la solution optimale $\hat{x}^k$ puis poser $l^k(\beta) = u^k(\beta) = \gamma^T \hat{x}^k$. Initialiser une liste $\mathcal{L}$ par $\mathcal{L} = \{\beta\}$, qui contiendra toutes les valeurs de β pour lesquelles on connaît la valeur exacte de la fonction valeur.

Etape 2 : Sélectionner *tous* les $x \in \mathbb{Z}_+^n$ tels que $x \leq \hat{x}^k$ pour leur appliquer la propriété (4.3.5) : si $Gx \in \mathbf{B}$, poser $l^k(Gx) = u^k(Gx) = \gamma^T x$, et si $\beta^k - Gx \in \mathbf{B}$, poser $l^k(\beta^k - Gx) = u^k(\beta^k - Gx) = \gamma^T(\hat{x}^k - x)$.

Etape 3 : Utiliser la propriété 4.3.3 pour mettre à jour bornes inférieures et supérieures. Par exemple, si $\beta' \leq \beta$ avec $\beta \in \mathcal{L}^k$, alors $l^k(\beta) \leftarrow \max\{l^k(\beta), l^k(\beta')\}$.

Etape 4 : Utiliser la super-additivité (propriété 4.3.4) pour mettre à jour d'autres bornes. Par exemple, si $\beta' \in \mathcal{L}^k$ mais $\beta \notin \mathcal{L}^k$, si $\beta + \beta' \in \mathbf{B} \backslash \mathcal{L}^k$, alors $u^k(\beta) \leftarrow \min\{u^k(\beta), u^k(\beta + \beta') - l^k(\beta)\}$

Etape 5 : Si toutes les valeurs de $z(\beta)$ sont connues, terminer. Sinon, aller à l'étape 1.

Démonstration. Il s'agit uniquement d'une énumération « améliorée » d'un nombre fini de problèmes. Dans le pire des cas, l'algorithme résout un à un ces problèmes, et ce en temps long mais fini. □

4.3.4 Résolution par branch and bound et résultats

L'algorithme de branch and bound est similaire à celui utilisé par Ahmed *et al.* (2004) (section 4.2). Le branchement est néanmoins plus simple : on choisit arbitrairement un côté de l'hyper-rectangle courant que l'on sépare en deux parties de même taille.

Les auteurs ont testé le problème sur des instances de différentes tailles. La plus grosse comporte 300 000 scénarios environ (l'algorithme est peu sensible au nombre de scénarios), avec 1000 variables de décision primaires, 500 variables de décision secondaires, mais seulement 7 contraintes secondaires (et aucune contrainte primaire).

En utilisant la programmation dynamique pour évaluer les fonctions valeurs (qui est plus rapide pour les petits problèmes, mais les auteurs s'attendent à ce que la méthode décrite ci-dessus soit plus rapide pour les gros problèmes) puis un algorithme de branch and bound et quelques améliorations, leur algorithme a résolu le problème en environ 23 heures. La résolution de la relaxation continue du problème par la méthode L-shaped a pris 121 secondes.

Etant très sensible au nombre m_2 de contraintes du problème secondaire, cette méthode semble peu adaptée à notre problème. Néanmoins, les auteurs rapportent que diverses implémentations récentes laissent à espérer qu'il est possible d'améliorer grandement l'efficacité de cet algorithme. Cela reste toujours du domaine de la recherche.

4.4 Utilisation des *test-sets*

Hemmecke et Schultz (2003) présentent une approche radicalement différente de celles précédemment exposées pour résoudre les problèmes stochastiques en nombres entiers, s'inspirant des algorithmes d'augmentation comme ceux permettant de résoudre les problèmes de flot (flot maximal ou flot de coût minimal). De manière analogue

aux chemins augmentants, ils utilisent des *test-sets* de Graver (1975), ensembles de vecteurs « améliorants », afin d'améliorer itérativement une solution réalisable jusqu'à l'optimalité.

Les test-sets de Graver sont certes de taille finie, mais très grande. Néanmoins, la grande similarité entre les différents sous-problèmes d'un problème stochastique en nombres entiers permet de les décomposer et ainsi de les générer plus facilement. Nous n'entrerons pas dans les détails de la procédure mais en présentons les grandes lignes.

4.4.1 Définitions

On considère la famille de problèmes en nombre entiers suivante :

$$(IP)_{c,b}: \ \min\{c^T z \mid Az = b, \ z \in \mathbb{Z}^n_+\} \tag{4.44}$$

On pose alors la définition suivante :

Définition 4.4.1 (Test-set)**.** *Un ensemble $\mathcal{T}_c \subseteq \mathbb{Z}^n$ est appelé* test-set *pour la famille de problèmes $(IP)_{c,b}$, b variant dans $\mathbb{R}^m$, si et seulement si :*

1. $\forall t \in \mathcal{T}_c, \ c^T t > 0$ et
2. *Pour tout $b \in \mathbb{R}^m$ et pour toute solution réalisable* non optimale *$z_0 \in \mathbb{Z}^n_+$ telle que $Az_0 = b$, il existe un vecteur $t \in \mathcal{T}_c$ tel que $z_0 - t$ est réalisable. Un tel vecteur est appelé un vecteur ou une direction améliorant(e).*

Un ensemble $\mathcal{T}$ est dit test-set *universel* pour la famille de problèmes $(IP)_{c,b}$ où c et b varient si pour tout $c \in \mathbb{R}^n$ il contient un test-set $\mathcal{T}_c$.

4.4.2 Principe et résultats

Il existe des tests-sets de cardinalité infinie, basés entre autres sur $\ker A$. En revanche, Graver (1975) a introduit des test-sets de cardinalité *finie*, appelés désormais *test-sets de Graver*. Avec un test-set de cardinalité finie, on peut utiliser l'algorithme d'augmentation 3.

Algorithme 3 Algorithme d'augmentation

Trouver un test-set de cardinalité finie $\mathcal{T}_c$ pour $(IP)_{c,b}$.
Trouver une solution réalisable z_0 pour $(IP)_{c,b}$
while $\exists t \in \mathcal{T}_c$ t.q. $c^T t > 0$ et $z_0 - t$ réalisable **do**
  $s_0 \leftarrow z_0 - t$
end while
RETURN z_0

Afin de trouver rapidement une solution réalisable, les auteurs montrent que l'on peut utiliser le test-set déjà construit. Le calcul du test-set est également accéléré en supposant que les matrices T et W de la contrainte secondaire $W(\omega)y = h(\omega) - T(\omega)x$ sont déterministes (et donc identiques quel que soit le scénario). En remarquant qu'un élément de $\ker A$ dont on a permuté certains « blocs » (voir équation 3.28) reste dans $\ker A$, ils en déduisent une manière de générer un test set $\mathcal{H}_\infty$, s'inspirant du test-set de Graver tout en étant bien plus rapide à évaluer. Ils prouvent en outre que leur test-set $\mathcal{H}_\infty$ est de cardinalité finie.

L'instance sur laquelle a été testé la méthode est petite (deux variables primaires, 4 variables secondaires). Elle comporte en revanche un grand nombre de scénarios (jusqu'à 35 721), les auteurs ayant remarqué que leur approche était peu sensible au nombre de scénarios. Sur cette instance, il faut 3.3 secondes pour calculer $\mathcal{H}^\infty$, et jusqu'à 180 secondes (cas où le nombre de scénarios est maximal) pour effectuer l'algorithme d'augmentation. `CPLEX`, quant à lui, ne parvient pas à résoudre en moins de 24 heures ce dernier problème.

Cette méthode, pourtant élégante, ne semble pas adaptée à notre cas. En effet, même si la possibilité d'inclure beaucoup de scénarios dans notre résolution est intéressante, le nombre de variables de décision est bien trop grand pour calculer entièrement $\mathcal{H}_\infty$.

Conclusion du chapitre : Après avoir présenté les différentes approches existant à l'heure actuelle permettant la résolution exacte de PSNE et en ayant étudié les résultats numériques de ces dernières, nous constatons qu'aucune ne semble adaptée à notre problème. A défaut d'inventer une nouvelle méthode de résolution des PSNE,

nous devrons nous contenter de développer, dans la partie suivante, une heuristique de résolution spécifique à notre problème.

Chapitre 5

RÉSOLUTION DU PROBLÈME

5.1 Première étude

5.1.1 Formulation et description des instances

Nous abordons maintenant la résolution du problème d'affectation de personnel présenté dans la première partie. Nous ajoutons en outre la possibilité d'avoir une sous-couverture SC_p^ω (pénalisée par un coût *csc*). Le problème se formule ainsi (les notations du problème sont présentées à la page 7) :

$$(P) \quad \min \quad \sum_{j\in J} c_j S_j + \sum_{\omega\in\Omega} p_\omega \left[\sum_{j\in JP} cp_j SP_j^\omega + \sum_{h\in H} cs_h SH_h^\omega + \sum_{p\in P} csc\ SC_p^\omega \right] \tag{5.1}$$

$$\text{slc} \quad \sum_{j\in J} A_{jp} S_j + \sum_{j\in JP} AP_{jp} SP_j^\omega - \sum_{k\in K} AK_{kp} B_k^\omega \tag{5.2}$$
$$+ \sum_{h\in H} AH_{hp} SH_h^\omega + SC_p^\omega \geq D_p^\omega,\ \forall(\omega,p) \in (\Omega, P)$$

$$\sum_{k\in K} Q_{jk} X_{jk}^\omega - S_j = 0,\ \forall(\omega,j) \in (\Omega, J) \tag{5.3}$$

$$\sum_{j\in J} Q_{jk} X_{jk}^\omega - B_k^\omega = 0,\ \forall(\omega,k) \in (\Omega, K) \tag{5.4}$$

$$\sum_{h\in H} R_{jh} X_{jh}^\omega \leq S_j,\ \forall(\omega,j) \in (\Omega, J) \tag{5.5}$$

$$\sum_{j\in J} R_{jh} XH_{jh}^\omega - SH_h^\omega = 0,\ \forall(\omega,h) \in (\Omega, H) \tag{5.6}$$

$$S_j,\ SP_j^\omega,\ SH_h^\omega,\ B_k^\omega,\ X_{jk}^\omega, XH_{jh}^\omega \in \mathbb{Z}^+, \tag{5.7}$$
$$SC_p^\omega \in \mathbb{R}^+,\ \forall(j,\omega,h,k) \in (J,\Omega,H,K)$$

Les instances du problème étudiées sont décrites ci-dessous. Les chiffres utilisés ont été choisis arbitrairement.

Périodes : On étudie une journée de 24 heures composée de 96 périodes discrètes de 15 minutes.

Scénarios : Leur nombre varie suivant les instances. Ils sont considérés comme équiprobables. Ils sont issus du travail de Legrain (2011), dont le sujet de maîtrise consistait à trouver des moyens d'évaluer efficacement la demande en employés. Le point de départ est une demande prévisionnelle fixée, à laquelle on applique des perturbations de manière aléatoire afin de générer autant de scénarios que l'on veut. Pour chaque demande prévisionnelle étudiée, on dispose ainsi de 10 000 scénarios que l'on considèrera comme décrivant *l'ensemble* des évènements réalisables.

Quarts : Chaque quart a une durée fixe de 8 heures. Il comporte une fenêtre de pause d'une heure et demie en son milieu. Tous les quarts possibles sont envisagés. Il y en a 65. Coût horaire : 1 (la demi-heure de pause n'est pas payée).

Pauses : Chaque pause dure une demi-heure. Chaque quart régulier doit inclure exactement une pause dans sa fenêtre de pause, et zéro ailleurs. Il y en a 94.

Temps partiels : Chaque quart à temps partiel dure 3 ou 4 heures. Tous les quarts à temps partiel possibles sont envisagés. Il y en a 166. Coût horaire : 1.25.

Heures supplémentaires : Il est possible d'ajouter une ou deux heures supplémentaires *après* un quart régulier. Coût horaire : 1.5.

Sous-couverture : Elle est autorisée. Coût horaire : 2.

Le tableau 5.1 indique la taille du problème en fonction du nombre de scénarios. Le solveur MIP `CPLEX` 12 peinant à résoudre des problèmes en nombres entiers comportant plus de quelques centaines de milliers de variables, on peut s'attendre à ce qu'il soit difficile de résoudre notre problème uniquement avec ce dernier dès lors que l'on dépasse les 25 scénarios.

Durant toute l'étude, nous avons utilisé le solveur `CPLEX` 12 d'IBM appelé depuis `Java`. Le langage `C++` n'a pas été utilisé pour des raisons de compatibilité avec d'autres outils existants au GERAD. De plus, les étapes les plus longues des algorithmes étant les résolutions de problèmes MIP par `CPLEX`, le gain de rapidité apporté par `C++` - résidant principalement dans la rapidité de modélisation - est assez faible.

# Scénarios	# Variables	# Contraintes
25	462 012	12 800
100	1 847 865	51 200
200	3 695 665	102 400
500	9 239 065	256 000
1000	18 478 065	512 000

Tableau 5.1 Taille du problème

Tous les calculs sont effectués sur un cluster `Sungrid` de PC sous `Linux` tous équipés de 1 ou 2 processeurs dual-core `AMD Opteron 275` à 2.2 GHz et équipés de 8 Go de RAM. Pour que tous les résultats soient tous comparables, nous avons interdit à `CPLEX` d'utiliser le multithreading.

Enfin, le logiciel `MATLAB` est utilisé pour exploiter les résultats.

5.1.2 Résolution LP

En premier lieu, et afin d'obtenir de premières informations sur le problème, nous avons étudié la résolution de sa relaxation continue. Dans toute cette partie, on a utilisé la même demande prévisionnelle, visualisable à la figure figure 5.1, accompagnée de son jeu de 10 000 scénarios. Lorsque rien d'autre n'est précisé, les scénarios sélectionnés pour la résolution sont les premiers de notre jeu. Ce dernier étant généré de manière aléatoire, choisir systématiquement les premiers scénarios n'introduit pas de biais.

Résolution par `CPLEX`

La première étape que nous avons effectuée est de résoudre la relaxation continue duproblème avec `CPLEX`, en utilisant directement la formulation (5.1)-(5.8). L'intérêt est de se faire une idée du temps nécessaire à la résolution du problème, et surtout d'obtenir de manière certaine la solution optimale de notre instance, pour ensuite vérifier que les solutions obtenues par d'autres méthodes sont bien correctes.

Le tableau 5.2 présente les résultats du calcul. La figure 5.1 présente la solution obtenue, représentée par l'évolution, en fonction du temps, du nombre d'employés

# Scénarios	25	100	200	500	1000
Valeur	447.823	448.838	448.466	450.460	451.009
Temps (sec)	14	375	1 415	17 085 (4h45)	28.5 h

TABLEAU 5.2 Résolution LP par `CPLEX`

demandés et de la couverture assurée par les différentes décisions possibles. Il s'agit d'une solution *cumulative* et *moyennée*. A l'exception de la courbe verte, représentant les quarts réguliers et ne variant pas en fonction du scénario, toutes les autres courbes sont moyennées sur la totalité des scénarios.

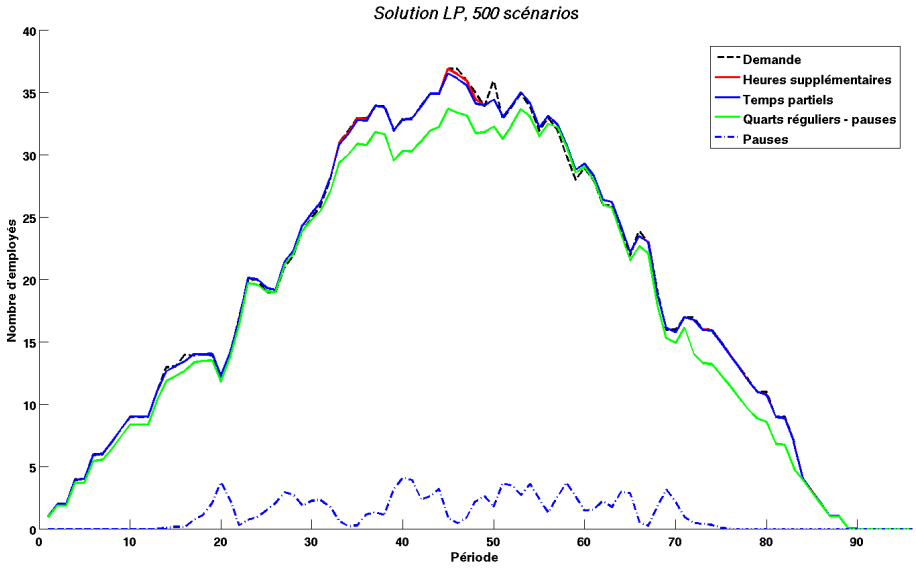


FIGURE 5.1 Solution LP pour 500 scénarios (moyenne)

On constate que si `CPLEX` est efficace, comme prévu, lorsqu'il y a peu de scénarios, il devient rapidement très lent. Rappelons en outre que nous ne calculons ici que la solution LP, et qu'il faudra ensuite prévoir un processus de branchement. Des temps dépassant les 1 000 secondes pour une seule résolution LP mèneront à des processus de branchement bien trop longs (même s'il est vrai que `CPLEX`, lors de son processus de branch and bound, garde en mémoire les bases du simplexe déjà utilisées, ce qui en accélère la résolution).

Résolution par la méthode L-shaped

Première formulation : La méthode *L-shaped* (voir algorithme 1, page 16) est, rappelons-le, une adaptation de la décomposition de Benders aux problèmes stochastiques avec recours. Notons également que, grâce à l'introduction de la sous-couverture, le problème secondaire est systématiquement réalisable. Il n'y a donc que des coupes *d'optimalité* à rajouter.

Le problème maître à résoudre est :

$$(P) \quad \min \quad \sum_{j\in J} c_j S_j + \sum_{\omega\in\Omega} p^\omega \theta^\omega \tag{5.8}$$

$$\text{slc} \quad \text{Coupes d'optimalité}$$

$$\forall j \in J,\ S_j \geq 0, \forall \omega \in \Omega,\ \theta^\omega \in \mathbb{R} \tag{5.9}$$

Les sous-problèmes s'écrivent (où l'on a indiqué pour chaque contrainte couplante la lettre grecque représentant sa variable duale associée) :

$$(SP)^\nu_\omega \quad \min \quad \sum_{j\in JP} cp_j SP^\omega_j + \sum_{h\in H} cs_h SH^\omega_h + \sum_{p\in P} cscSC^\omega_p \tag{5.10}$$

$$\text{slc} \quad \sum_{j\in J} A_{jp} S^\nu_j + \sum_{j\in JP} AP_{jp} SP^\omega_j - \sum_{k\in K} AK_{kp} B^\omega_k \tag{5.11}$$

$$+ \sum_{h\in H} AH_{hp} SH^\omega_h + SC^\omega_p \geq D^\omega_p,\ \forall p \in P \rightarrow \alpha_p$$

$$\sum_{k\in K} Q_{jk} X^\omega_{jk} - S^\nu_j = 0,\ \forall j \in J\ \rightarrow \beta_j \tag{5.12}$$

$$\sum_{j\in J} Q_{jk} X^\omega_{jk} - B^\omega_k = 0,\ \forall k \in K \tag{5.13}$$

$$\sum_{h\in H} R_{jh} X^\omega_{jh} \leq S^\nu_j,\ \forall j \in J \rightarrow \gamma_j \tag{5.14}$$

$$\sum_{j\in J} R_{jh} XH^\omega_{jh} - SH^\omega_h = 0,\ \forall h \in H \tag{5.15}$$

$$SP^\omega_j,\ SH^\omega_h,\ B^\omega_k,\ X^\omega_{jk},\ XH^\omega_{jh} \in \mathbb{R}^+,\ \forall (j,h,k) \in (J,H,K) \tag{5.16}$$

Remarque 5.1.1. *Nous utilisons la même convention duale que* `CPLEX` *: toutes les*

variables associées à des contraintes d'inégalité sont choisies négatives.

Pour chaque sous-problème $(SP)^{\nu}_{\omega}$ dont les solutions duales optimales sont $(\alpha^*,\ \beta^*,\ \gamma^*)$, la coupe d'optimalité associée s'écrit :

$$\theta^{\omega} + \sum_{p \in P} \sum_{j \in J} \alpha^*_p A_{jp} S_j - \sum_{j \in J} \left(\beta^*_j + \gamma^*_j\right) S_j \geq \sum_{p \in P} \alpha^*_p D^{\omega}_p \tag{5.17}$$

A chaque itération de la méthode L-shaped, on ajoute une coupe d'optimalité par scénario. L'algorithme est programmé en `Java` et `CPLEX` 12 est utilisé pour résoudre les différents problèmes linéaires rencontrés lors de son exécution.

Remarque 5.1.2. *Pour obtenir les variables duales qui nous intéressent, on force `CPLEX` à résoudre le problème secondaire en utilisant la méthode du simplexe dual. On peut alors, à la fin de la résolution, obtenir les variables duales désirées.*

Dans cette formulation simple, on constate que les variables du problème maître sont complètement libres : ce sont en fait les problèmes secondaires qui « font » tout le travail. En particulier, lors des premières itérations de la méthode L-shaped, la solution maître risque d'aller dans n'importe quelle direction, car trop libre.

Nous avons donc étudié une seconde formulation du problème qui, au prix d'une augmentation de la complexité du problème esclave, va mieux cadrer le problème maître. L'espoir est ainsi de diminuer le nombre d'itérations nécessaires à la résolution du problème.

Seconde formulation : L'idée de cette formulation est de ramener la contrainte de couverture dans le problème maître. Pour cela, on ramène l'achat de sous-couverture dans ce dernier. Cette sous-couverture est la « pire » possible : on considère dans un premier temps que les quarts réguliers sont en pause pendant *toute* leur fenêtre de pause, et on demande à ce que la somme des personnes travaillant sur des quarts réguliers et de la sous-couverture achetée soit supérieure à la demande.

Le problème secondaire va alors « racheter » toute la sous-couverture achetée pour rien, c'est-à-dire celle compensée par :

– Les quarts réguliers qui ne sont *pas* en pause, alors qu'ils pourraient.

– Les quarts partiels achetés.
– Les heures supplémentaires achetées.

On introduit les paramètres f_{jp} définis par :

$$\forall j \in J, p \in P,\ f_{jp} = \begin{cases} 1 \text{ si la fenêtre de pause du quart } j \text{ recouvre } p \\ 0 \text{ sinon} \end{cases} \tag{5.18}$$

Le problème maître s'écrit alors :

$$(P') \quad \min \quad \sum_{j \in J} c_j S_j + \sum_{\omega \in \Omega} p^\omega \sum_{p \in P} SC_p^\omega + \sum_{\omega \in \Omega} p^\omega \theta^\omega \tag{5.19}$$

$$\text{slc} \quad \sum_{j \in J} (A_{jp} - f_{jp})\, S_j + SC_p^\omega \geq D_p^\omega,\ \forall (\omega, p) \in (\Omega, P) \tag{5.20}$$

$$\text{Coupes d'optimalité} \tag{5.21}$$

$$\forall j \in J,\ S_j \geq 0, \forall \omega \in \Omega,\ \theta^\omega \in \mathbb{R} \tag{5.22}$$

Notons RSC_p^ω les variables secondaires de remboursement. Les problèmes secondaires s'écrivent de la manière suivante (comme précédemment, on indique les variables duales associées aux contraintes couplantes) :

$$(SP')_\omega^\nu \quad \min \quad -\sum_{p \in P} cscRSC_p^\omega + \sum_{j \in JP} cp_j SP_j^\omega + \sum_{h \in H} cs_h SH_h^\omega \tag{5.23}$$

$$\text{slc} \quad RSC_p^\omega \leq \sum_{j \in J} f_{jp} S_j^\nu - \sum_{k \in K} AK_{kp} B_k^\omega + \sum_{j \in JP} AP_j SP_j^\omega \tag{5.24}$$

$$+ \sum_{h \in H} AH_{hp} SH_h^\omega,\ \forall p \in P \rightarrow \delta_p$$

$$RSC_p^\omega \leq SC_p^\omega,\ \forall p \in P \rightarrow \epsilon_p \tag{5.25}$$

$$\sum_{k \in K} Q_{jk} X_{jk}^\omega - S_j^\nu = 0,\ \forall j \in J\ \rightarrow \zeta_j \tag{5.26}$$

$$\sum_{j \in J} Q_{jk} X_{jk}^\omega - B_k^\omega = 0,\ \forall k \in K \tag{5.27}$$

$$\sum_{h\in H} R_{jh}X^{\omega}_{jh} \leq S^{\nu}_{j},\ \forall j \in J\ \rightarrow \eta_j \tag{5.28}$$

$$\sum_{j\in J} R_{jh}XH^{\omega}_{jh} - SH^{\omega}_{h} = 0,\ \forall h \in H \tag{5.29}$$

$$SP^{\omega}_{j},\ SH^{\omega}_{h},\ B^{\omega}_{k},\ X^{\omega}_{jk},\ XH^{\omega}_{jh} \in \mathbb{R}^{+},\ \forall (j,h,k) \in (J,H,K) \tag{5.30}$$

La première contrainte impose que l'on ne rembourse pas plus de sous-couverture que l'on en a « racheté » dans le problème secondaire (heures supplémentaires, temps partiels, pauses non effectuées). La contrainte (5.25) impose que l'on ne rembourse jamais plus de sous-couverture qu'on en a acheté dans le problème maître, comme cela pourrait se produire en cas de sur-couverture.

La coupe d'optimalité associée à chaque problème $(SP')^{\nu}_{\omega}$ s'écrit :

$$\theta^{\omega} + \sum_{p\in P}\left(\sum_{j\in J}\delta^{*}_{j}f(j,p)S_j - \epsilon^{*}_{p}SC^{\omega}_{p}\right) - \sum_{j\in J}\left(\zeta^{*}_{j} + \eta^{*}_{j}\right)S_j \geq 0 \tag{5.31}$$

Résultats : Les tableaux 5.3-5.4 présentent les résultats des calculs pour la première et seconde formulation respectivement.

# Scénarios	25	100	200	500	1 000
# Itérations	189	102	93	87	82
Valeur	447.823	448.838	448.466	450.460	451.009
Temps (sec.)	100	387	867	2 667	9 453 (2h40)

Tableau 5.3 Résultats pour la première formulation

# Scénarios	25	100	200	500	1 000
# Itérations	108	67	64	57	61
Valeur	447.823	448.838	448.466	450.460	451.009
Temps (sec.)	130	633	1 743	9 595	53 600 (15h)

Tableau 5.4 Résultats pour la seconde formulation

Remarque 5.1.3. *Même si les résultats sont présentés avec une précision de* 10^{-3}*, les résultats fournis par la résolution LP par* `CPLEX` *et par la méthode L-shaped sont égaux à* 10^{-11} *près.*

On constate que même si la seconde formulation a pour effet de diminuer le nombre d'itérations L-shaped nécessaires à la résolution du problème, l'augmentation de la complexité des deux problèmes l'emporte, en terme de vitesse, sur les quelques itérations gagnées quel que soit le nombre de scénarios. La première formulation est donc dans tous les cas plus intéressante que la seconde.

Comparaison

La figure 5.2 indique, sur une échelle logarithmique, les différents temps de résolution pour chaque méthode en fonction du nombre de scénarios. Pour un faible nombre de scénarios (25), `CPLEX` est nettement plus rapide que la méthode L-shaped, et il y a donc peu d'espoir de trouver un algorithme encore plus rapide. En revanche, dès que l'on dépasse les 100 scénarios, la méthode L-shaped est plus rapide. L'écart va en s'amplifiant au fur et à mesure que l'on augmente le nombre de scénarios.

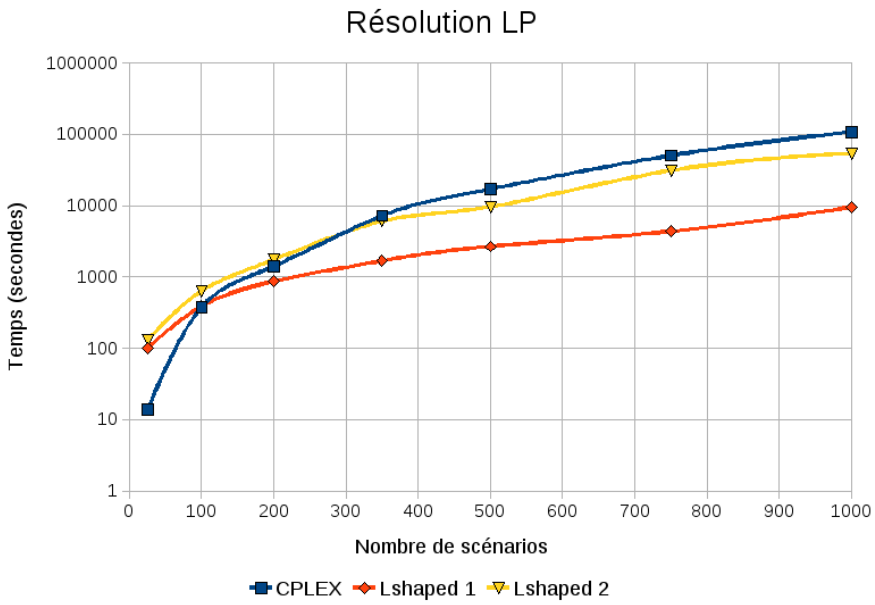


FIGURE 5.2 Temps de résolution LP, échelle logarithmique

Le bilan de cette étude préliminaire est globalement bon. La méthode L-shaped est nettement plus rapide que `CPLEX` lorsqu'il y a un grand nombre de scénarios, elle est en revanche beaucoup plus lente que lui lorsqu'il y en a peu. En outre, même

si il faut 10 fois moins de temps à la méthode L-shaped pour résoudre l'instance à 1000 scénarios (20 millions de variables), le temps nécessaire à cette résolution reste a priori trop élevé pour envisager un algorithme de branch and bound exact.

Notre problème étant un problème industriel, il faut dès lors faire preuve de pragmatisme. Il ne sera a priori pas possible, le temps de cette maîtrise, de trouver un algorithme miracle résolvant systématiquement le problème de manière exacte. Il nous faut donc sélectionner la difficulté sur laquelle nous allons porter nos efforts :

1. Trouver une solution exacte, au détriment du nombre de scénarios.
2. Augmenter le nombre de scénarios, quitte à utiliser une heuristique pour la résolution.

Pour établir un critère de sélection, rappelons-nous que le but du problème avec recours est de trouver une meilleure solution que la solution dite *moyenne*, qui ne prend la décision primaire qu'en fonction de l'espérance de la demande, et qui adapte par la suite son recours. Le gain[1] est appelé la *valeur de la solution stochastique*. Intuitivement, plus le nombre de scénarios est élevé, plus ce gain est grand. En revanche, l'utilisation d'une heuristique pour la résolution du problème IP fait perdre une partie de ce gain. La méthode retenue sera donc celle qui permet de « capturer », en moyenne, la plus grosse partie de ce gain.

5.1.3 Comparaison : approches déterministe et « wait and see »

Afin de choisir l'une ou l'autre des pistes d'amélioration, il nous faut pouvoir comparer la qualité des solutions obtenues. Pour cela, nous nous baserons sur « l'amélioration » qu'apportera chaque méthode par rapport à la solution moyenne. Nous allons préciser ce dont il s'agit et comment nous la calculons.

Valeur des solutions moyennes et « wait and see »

En toute généralité, et en notant x la variable de décision primaire, notre problème stochastique s'écrit sous la forme suivante :

[1] Il reste à le définir précisément, entre autres faire la différence entre la valeur du problème résolu sur N scénarios et la valeur de la solution testée ensuite sur les 10 000 scénarios.

$$\min_x \mathbb{E}_\xi \left(z(x, \xi) \right) \tag{5.32}$$

En notant $\bar{\xi} = \mathbb{E}(\xi)$, le problème *moyen* est le suivant :

$$(EV)\ \bar{x}(\bar{\xi}) = \min_x z(x, \bar{\xi}) \tag{5.33}$$

On appelle espérance de la solution moyenne la quantité :

$$EEV = \mathbb{E}_\xi \left[z(\bar{x}(\bar{\xi}), \xi) \right] \tag{5.34}$$

La quantité EEV est rapide à calculer, mais elle n'est pas optimale. Comme écrit dans la première partie, calculer la solution moyenne (dite également *déterministe*) revient à ignorer l'information sur la distribution de la variable aléatoire pour n'en garder que la moyenne : c'est une perte d'information.

Afin de calculer EEV pour le premier jeu de données, nous procédons de la manière suivante. Nous résolvons d'abord le problème en utilisant l'estimation qui a servi à construire les scénarios : nous résolvons en pratique le même problème que précédemment, en utilisant comme unique scénario cette demande prévisionnelle. Nous conservons uniquement la solution du problème « maître », notée $\bar{S}$.

Ensuite, nous résolvons[2], pour les 10 000 scénarios qui constituent notre jeu de données, le problème secondaire, en utilisant comme solution primaire $\bar{S}$. En supposant que les 10 000 scénarios représentent de manière correcte l'intégralité des évènements aléatoires possibles, nous obtenons une estimation de EEV en prenant l'espérance du coût de ces 10 000 problèmes. Concrètement, notre méthode correspond à utiliser la demande prévisionnelle afin de déterminer les quarts réguliers, puis à effectuer au jour le jour des ajustements grâce aux pauses, aux heures supplémentaires et aux temps partiels.

[2]En LP pour l'instant

Nous avons en outre calculé la valeur de la solution *wait and see* (notée WS). Elle consiste à n'attribuer les quarts réguliers qu'une fois que l'on connaît la demande, ce qui revient à effectuer une attribution de quarts réguliers par scénario (soit à découper notre problème en $|\Omega|$ problèmes indépendants). On peut démontrer que, pour un nombre de scénarios *fixé*, on a l'inégalité suivante :

$$WS \leq RP \leq EEV \tag{5.35}$$

où RP désigne le coût de la solution du problème *avec recours* (soit celui que l'on tente de résoudre). Les valeurs pour les problèmes WS et EEV sont décrites dans le tableau 5.5.

WS	EEV
447.123	455.368

Tableau 5.5 Valeur EEV et WS, 10 000 scénarios, demande « standard »

Valeur des solutions stochastiques

Afin de comparer nos différentes méthodes de résolution nous calculons, pour chaque solution maître, son coût *réel*. En effet, le coût seul d'une solution stochastique, avec un nombre de scénarios déterminés, est peu significatif pour nous. Par exemple, si l'on ne sélectionne que 2 ou 3 scénarios de notre série de 10 000 (la série étant supposée décrire tous les scénarios possibles), il est probable que l'on obtienne une solution moins coûteuse a priori que si l'on en sélectionne 500. Pourtant, lorsque testée dans la « vraie vie », *i.e.* sur 10 000 scénarios, il est également fort probable que la solution maître à 500 scénarios soit meilleure que celle à 2 ou 3 scénarios.

Afin d'évaluer le coût *réel* d'une solution stochastique, on procède de la façon suivante :

1. Résolution du problème stochastique avec un nombre de scénarios fixé et faible devant 10 000.
2. Calcul pour chacun des 10 000 scénarios envisageables du recours associé à la solution maître trouvée.

3. Calcul du coût moyen de la solution sur les 10 000 scénarios.

Pour commencer, nous avons calculé la valeur réelle de différentes instances, chacune constituée des N premiers scénarios de notre jeu, lorsque N varie. La comparaison entre les coûts des solutions WS, RP et EEV est présentée figure 5.3.

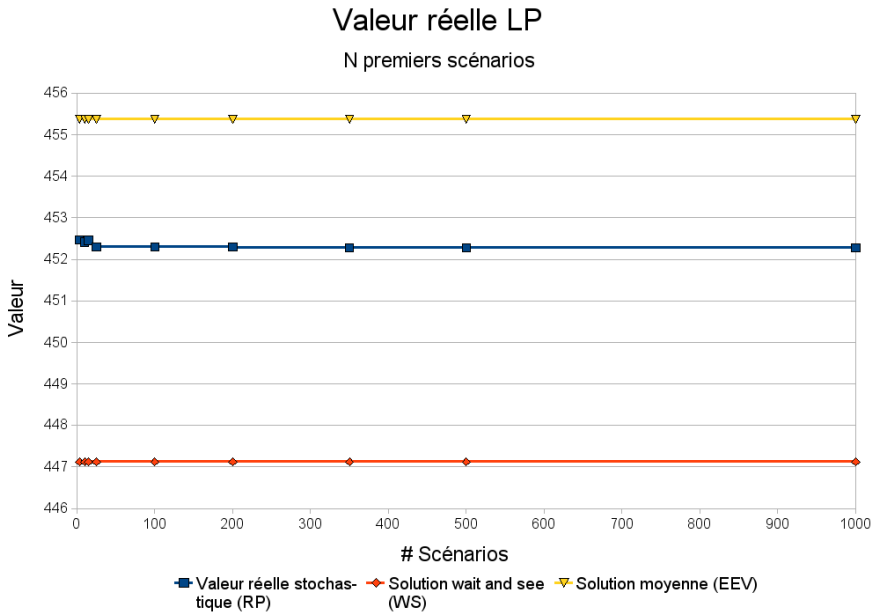
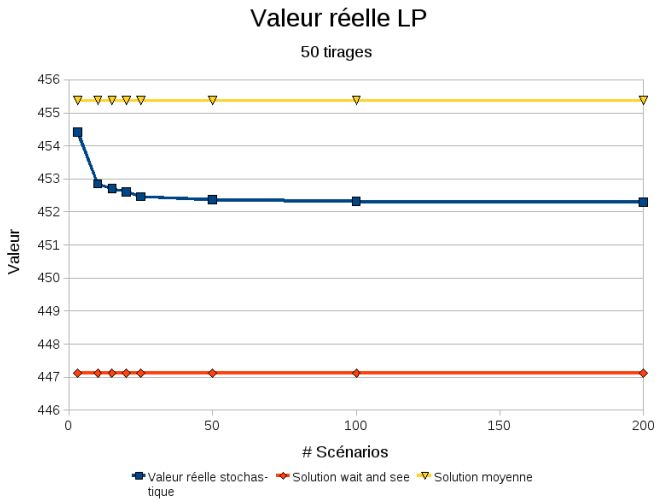


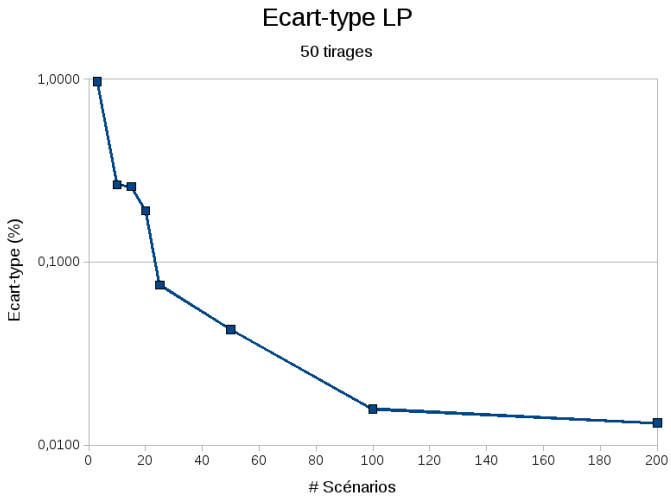
FIGURE 5.3 Valeurs réelles de solutions stochastiques

Au delà de 50 scénarios, les valeurs réelles des solutions semblent être à peu près constantes. Néanmoins, nous n'avons effectué ici qu'une seule évaluation par nombre de scénarios, et il se peut que les scénarios sélectionnés ne soient pas représentatifs.

Nous avons donc relancé ces calculs en résolvant systématiquement 50 instances à nombre de scénarios fixé N, et en tirant pour chaque instance les N scénarios de manière aléatoire dans notre banque de 10 000 scénarios. Par exemple, pour 20 scénarios, on sélectionne aléatoirement 50 ensembles de 20 scénarios, et l'on calcule ensuite la valeur réelle moyenne des problèmes ainsi générés ainsi que l'écart-type de cette valeur.



(a) Valeur réelle



(b) Ecart-type (échelle logarithmique)

# Scénarios	3	10	15	20
Valeur réelle	454.423	452.852	452.705	452.611
Ecart-type (%)	0.968	0.264	0.258	0.190
Δ sol. préc.	/	1.571	0.147	0.093

(c) Valeurs numériques (1)

# Scénarios	25	50	100	200
Valeur réelle	452.468	452.371	452.317	452.300
Ecart-type (%)	0.075	0.043	0.016	0.013
Δ sol. préc.	0.144	0.097	0.053	0.018

(d) Valeurs numériques (2)

FIGURE 5.4 Valeur réelle des solutions stochastiques - 50 tirages

La figure 5.4 présente les résultats. L'écart-type est presque aussi important que la moyenne. Gardons en effet à l'esprit que l'on ne souhaite résoudre *qu'une seule fois* le problème stochastique. Un écart-type trop élevé indiquerait que l'on obtiendrait des résultats très différents suivant les scénarios sur lesquels ont choisit d'optimiser, ce qui n'est pas du tout souhaitable.

Autres jeux de données

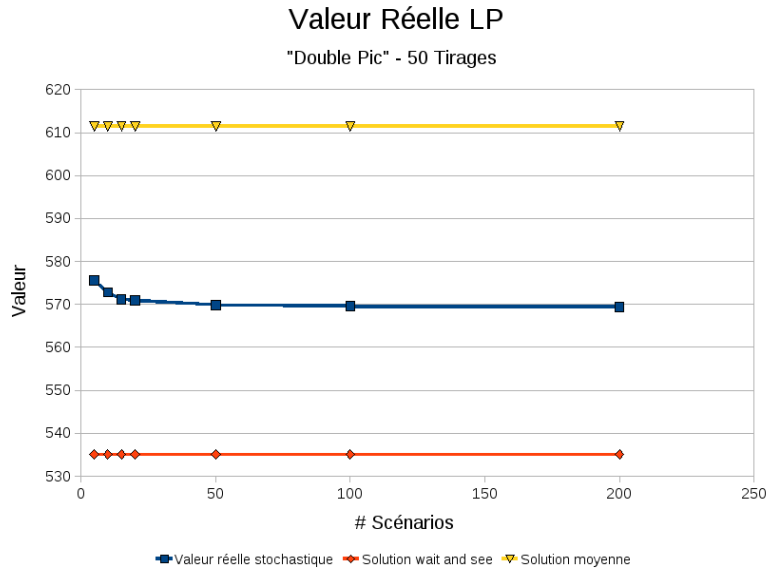
Afin de ne pas restreindre notre étude à un cas particulier, nous avons testé nos méthodes de résolution sur deux autres instances. La première, nommée « double pic » comporte 10 000 scénarios générés à partir d'une demande prévisionnelle présentant deux pics de demande. Les résultats sont présentés figure 5.5.

On constate sur la figure 5.5 que la variante « double pic » présente le même comportement que notre instance initiale. La valeur « réelle » des problèmes stochastiques diminue en moyenne lorsque le nombre de scénarios augmente. L'écart-type de la valeur de chaque tirage diminue également. Tout indique que notre méthode est appropriée ici, car elle fournit de bien meilleures solutions que la solution moyenne, et un écart-type faible entre les différents tirages assure une bonne stabilité.

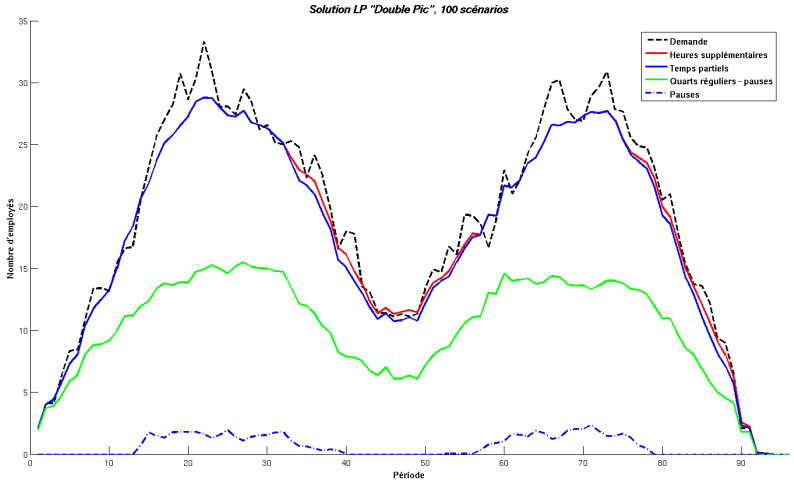
La seconde instance étudiée, nommée « dents de scie » utilise une demande prévisionnelle beaucoup plus irrégulière que les deux autres jeux. La figure 5.6 présente les résultats du calcul.

On ne constate pas de baisse régulière de la valeur moyenne de la solution stochastique, que ce soit avec 50 ou 200 tirages. Cela est dû à un écart-type trop important. Ainsi, les 2.14 % d'écart-type relatif pour les 200 scénarios à 50 tirages correspondent à un écart-type absolu de 12.29 : bien supérieur aux variations observées lorsque l'on augmente le nombre de scénarios.

En revanche, l'*intervalle de confiance* à 95% de la valeur réelle moyenne pour ces 200 scénarios et 50 tirages est $[572.06, 578.88]$. Le gain *moyen* par rapport à la solution moyenne est donc certain.



(a) Valeur réelle

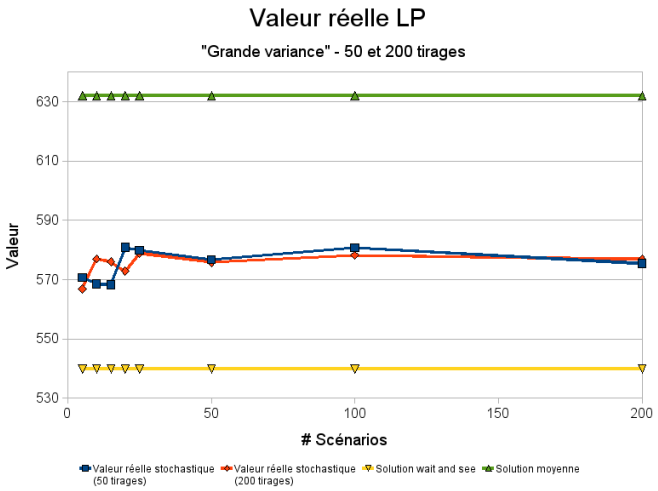


(b) Solution stochastique (moyenne)

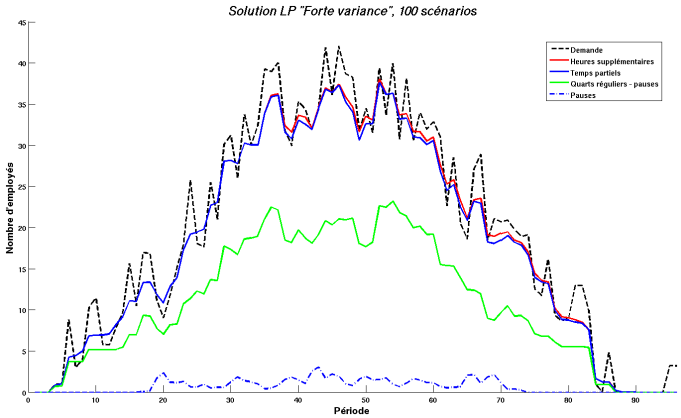
# Scénarios	5	10	20	50	100	200
Valeur réelle	575.584	572.820	570.934	569.890	569.588	569.400
Ecart-type (%)	0.71	0.35	0.15	0.05	0.03	0.02

(c) Valeurs numériques

FIGURE 5.5 Valeur réelle des solutions stochastiques - Double pic



(a) Valeur réelle



(b) Solution stochastique (moyenne)

# Scénarios	5	10	15	20
Valeur 50 tirages	570.670	568.593	568.315	580.843
Ecart-type 50 (%)	11.56	8.97	6.74	7.69
Valeur 200 tirages	566.876	577.023	576.006	572.832
Ecart-type 200 (%)	14.34	10.51	8.00	6.51

(c) Valeurs numériques (1)

# Scénarios	25	50	100	200
Valeur 50 tirages	579.868	576.788	580.788	575.468
Ecart-type 50 (%)	5.92	3.63	3.29	2.14
Valeur 200 tirages	578.926	575.813	578.207	576.933
Ecart-type 200 (%)	6.37	4.11	2.92	1.96

(d) Valeurs numériques (2)

FIGURE 5.6 Valeur réelle des solutions stochastiques - Dents de scie

On remarque en outre que le niveau d'emploi des quarts réguliers est plus faible pour ces deux jeux que pour le jeu standard.

Définition 5.1.1. *Nous appelons* écart-type relatif moyen *ou encore* coefficient de variation moyen *d'un jeu de scénarios la valeur γ telle que :*

$$\gamma = \frac{1}{P}\left[\sum_{p=1}^{P}\left(\frac{1}{\bar{D}_p}\frac{1}{K}\sqrt{\sum_{k=1}^{K}(D_p^k - \bar{D}_p)^2}\right)\right]$$

Il s'agit de la moyenne, sur chaque période p, du coefficient de variation (écart-type / moyenne) de la demande à la période p. Si pour une période p la demande moyenne est nulle, on considère que le terme associé à p dans la première somme est nul.

Pour les 10 000 scénarios de chaque jeu de données, nous avons calculé l'écart-type moyen par période. Les résultats du calcul sont présentés sur le tableau 5.6.

Jeu de données	Standard	Double pic	Dents de scie
Ecart-type	1.1312	12.8981	13.0017

Tableau 5.6 Écarts-type moyen des jeux de données

Conclusions, premiers résultats IP

Pour notre jeu de données de départ (figure 5.4), on remarque que 25 scénarios semblent être un bon compris pour obtenir une bonne solution stochastique entière avec `CPLEX` tout en gardant un temps de calcul raisonnable. En effet, l'écart-type trouvé pour nos 50 tirages à 25 scénarios est très faible (0.08 %) et la valeur moyenne des solutions semble être proche de l'asymptote apparaissant à la figure (5.3), à environ 452.3.

Nous avons donc résolu notre problème initial sur 25 scénarios en imposant l'intégrité des variables de quarts réguliers *seulement* (ces dernières ayant beaucoup plus de « poids » que les variables secondaires). Le calcul est effectué avec `CPLEX` seul (algorithme du simplexe et éventuellement de branch and bound), les résultats sont présentés dans le tableau 5.7.

	LP	MIP exact	MIP, tolérance 0.2%
Valeur sur 25 scénarios	447.823	448.200	448.385
Valeur réelle	452.300	452.635	452.759
Temps (sec)	14	21 689	27

TABLEAU 5.7 Résolutions par `CPLEX`, 25 scénarios

L'ajout d'une tolérance à 0.2% (très souvent effectuée en pratique), diminue très fortement le temps de résolution IP, signe que le problème comporte des symétries qu'il pourra être intéressant de briser ultérieurement.

Pour les autres jeux de données, les résultats sont plus mitigés. Dans tous les cas, la solution stochastique s'avère être bien meilleure que la solution moyenne : gain de l'ordre de 5 à 10%, à comparer au 1% de gain sur notre jeu initial. En revanche, si les scénarios présentent un trop grand écart-type relatif moyen, notre méthode perd grandement en stabilité.

En conclusion, nous retenons de cette première étude qu'il y a bel et bien à intérêt à chercher une solution stochastique plutôt qu'une solution moyenne. Dans tous les cas, le résultat semble meilleur. Une demande trop irrégulière aura en revanche tendance à rendre la méthode moins stable.

Afin d'améliorer la précision de notre méthode, nous choisirons d'augmenter le nombre de scénarios plutôt que de chercher à tout prix une solution exacte. En effet, il nous semble préférable de perdre un peu d'optimalité mais de gagner en stabilité. Nous augmenterons le nombre de scénarios jusqu'à 500.

5.2 Résolution IP

En premier lieu, nous avons résolu le problème avec `CPLEX` avec toutes les variables entières (sauf SC), pour 25 scénarios, avec une tolérance MIP de 0.2 %. La valeur de la solution était 452.213, à comparer à 448.385 obtenu lorsque seul le problème maître est entier. Il a fallu à CPLEX pour résoudre cela 7 000 secondes. Cela semble relativement long, d'autant plus que seulement 25 scénarios étaient envisagés.

Une amélioration rapide de la précision de cette résolution semblant être difficile à obtenir, nous sommes confortés dans notre idée, développée à la section 5.1.3, de porter nos efforts sur une augmentation jusqu'à 500 du nombre de scénarios plutôt que sur une résolution plus précise.

Pour s'en convaincre, on peut consulter la figure 5.7. Elle représente la part du coût du recours dans le coût total. Le calcul est effectué sur différentes instances (LP) à 500 scénarios via la méthode L-shaped utilisée précédemment.

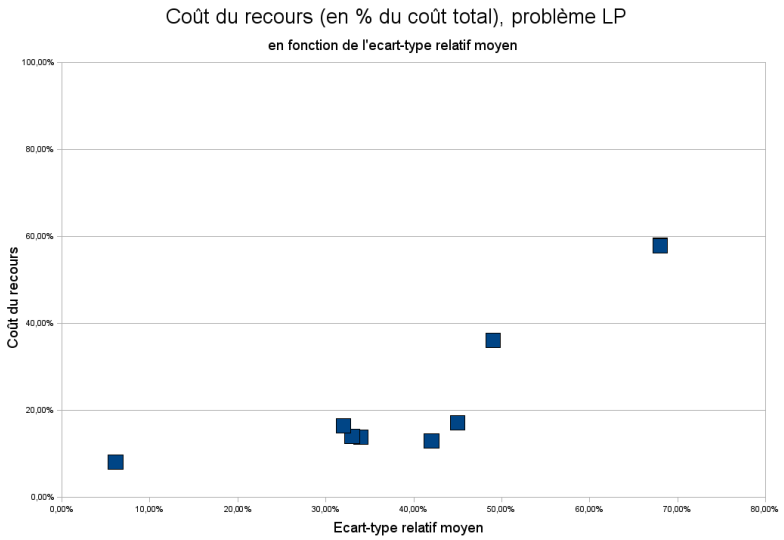


FIGURE 5.7 Importance du coût du recours dans le coût total

Dans la majorité des cas, cette part est inférieure à 20%. En outre, il ne faut pas oublier que le coût du recours est une *moyenne*. Ainsi, si l'on considère un modèle à 500 scénarios, chaque recours spécifique à un scénario ne représente que $\frac{1}{500}$-ième du coût du recours total. Un branchement sur une telle variable de décision n'aura donc qu'une importance très faible comparativement à un branchement sur une variable primaire.

5.2.1 Heuristique

Description

Nous avons utilisé comme base la méthode L-shaped décrite auparavant. A 500 scénarios, cette dernière est en effet nettement plus rapide que `CPLEX` (voir figure 5.2). Nous la combinons avec une heuristique classique de branchement, décrite ci-après et schématisée sur la figure 5.8.

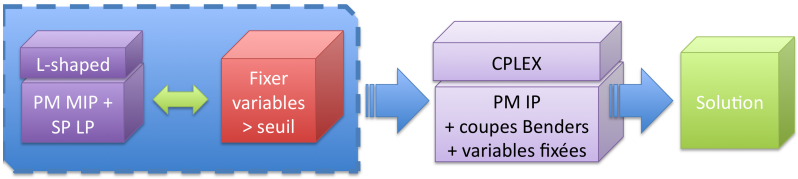


FIGURE 5.8 Description de l'heuristique

Le problème à 500 scénarios est d'abord résolu en LP par la méthode L-shaped. Ensuite, on fixe à la valeur entière supérieure *toutes* les variables dont la partie fractionnaire est supérieure à un seuil déterminé (typiquement : 0.8, valeur nous ayant permis d'obtenir les meilleurs résultats). On résout à nouveau le problème LP. On prend alors soin de conserver les coupes de Benders déjà calculées : elles sont toujours valides et permettent d'accélérer grandement la résolution du problème (facteur 10 et plus). S'il existe des variables non encore fixées dont la partie fractionnaire dépasse le seuil, on les fixe et on résout à nouveau le problème. On itère ainsi jusqu'à ce qu'il n'y ait plus aucune variable à fixer.

A la fin de ces itérations, on récupère un problème maître de la forme :

$$(PM) \quad \min \quad \sum_{j \in J} c_j S_j + \theta \tag{5.36}$$

$$\text{variables fixées} \tag{5.37}$$

$$\text{coupes de Benders} \tag{5.38}$$

Comme dit précédemment, la méthode L-shaped revient à minorer la fonction recours (représentée par θ) par des hyperplans. Elle s'arrête lorsque cette minoration

est une égalité pour la solution maître envisagée.

L'approximation que l'on fait ici va être d'utiliser la minoration trouvée par la méthode de Benders comme « vraie » valeur de θ. Pour cela, on résoud directement (PM) pour obtenir les quarts réguliers S_j, en conservant les coupes de Benders déjà accumulées.

Ce problème est tout à fait résoluble en IP par `CPLEX`. Il ne comporte en effet que peu de variables : seulement une par quart et une par scénario (soit 565 dans notre cas), à comparer avec la dizaine de millions du problème d'origine. Il faudra garder à l'esprit que la solution trouvée pourra avoir une valeur *inférieure* à l'optimum réel, car θ est une sous-estimation des coûts du recours.

Discussion et résultats attendus

Cette méthode ne calcule pas directement les solutions des problèmes secondaires, mais celles-ci ne sont pas importantes, si ce n'est pour l'évaluation du coût réel de la solution. En effet, les décisions importantes sont les décision primaires. Les décisions secondaires ne sont prises qu'au jour le jour, et sont individuellement très rapides à calculer dès que l'on a connaissance de la solution maître.

On peut également avancer que l'on perd en optimalité en résolvant les problèmes secondaires en LP dans notre méthode, alors qu'ils sont normalement en IP. Nous voyons deux arguments permettant de prédire que cela ne sera pas trop dérangeant :

- Le coût du problème secondaire est de l'ordre de 10% du coût total. On a vu précédemment que le gap d'intégrité était à priori faible. Même si ce dernier est de 1%, cela représente 0.1% environ du coût total : son importance est toute relative.
- Les 500 scénarios ont un effet « moyennant » qui diminue l'influence du gap d'intégrité.

En pratique, une telle heuristique de résolution sera intéressante si elle permet de récupérer au minimum 50% du gain qu'il existe entre solution moyenne et solution

stochastique. Elle sera réellement exploitable si elle permet d'en récupérer au minimum 75 %. Il reste néanmoins à savoir comment calculer ce gain : calculer la solution stochastique exacte en IP n'est évidemment pas envisageable.

5.2.2 Étude d'un modèle « jouet »

Afin d'évaluer la performance de notre méthode face à une résolution exacte, nous avons tout d'abord utilisé un modèle « jouet », comportant moins de variables que le problème de départ (de 100 000 à 800 000 environ). Nous avons résolu le problème stochastique en nombre entiers de manière exacte grâce à `CPLEX`, puis nous l'avons résolu avec notre heuristique. Nous analysons ensuite le coût réel des deux solutions obtenues afin de les comparer. Décrivons tout d'abord les instances utilisées :

Demande prévisionnelle : double pic

Quarts : durée de 8 heures, fenêtre de pause de 1 heure. Un quart toutes les heures (17 quarts au total)

Temps partiels : durée 3 heures. Un toutes les demi-heures (43 au total).

Heures supplémentaires : durée 1 heure. Une toutes les heures (16 au total).

Pauses : durée 30 minutes.

Le tableau 5.2.2 présente les résultats de nos calculs. La résolution exacte du problème est effectuée *sans* tolérance pour la méthode de branch and bound. L'*évaluation* du coût réel sur 10 000 scénarios est effectuée avec une tolérance de 0.5% *par scénario*. Sachant que le coût du recours est de l'ordre de 10% du coût total, cela correspond à une erreur de l'ordre de 0.05% du coût total.

On définit la *distance* entre les quarts *réguliers* S^0 d'une solution exacte et S^1 une solution heuristique par :

# Scénarios	Valeur sol. exacte	Valeur heur.	Écart (%)	Distance
100	581.986	581.986	0	0
200	581.686	581.680	-0.001	2
500	581.679	581.725	0.01	1

Tableau 5.8 Modèle jouet : résultats

$$\sum_{j \in J} |S_j^1 - S_j^0| \tag{5.39}$$

Deux solutions sont identiques si et seulement si leur distance est nulle.

On constate sur le tableau 5.2.2 que notre heuristique est bonne (et précise) pour ce jeu de données. En effet, l'écart le plus grand constaté, à 500 scénarios, est de 0.01% entre les deux solutions. Cet écart est négligeable. Il est en effet du même ordre que la tolérance utilisée lors de l'évaluation du coût réel IP. Il est également un ordre de grandeur plus faible que le flou introduit par la multiplicité des solutions moyennes (*cf.* section 5.3.1).

5.2.3 Protocole de test retenu

Valeurs calculées

Pour chaque résolution, nous utilisons les 500 *premiers* scénarios du jeu de données considéré. Pour chaque jeu de données et variante de l'heuristique considérées, nous calculons trois valeurs :

- La valeur « affichée » de l'heuristique (*i.e.* celle du problème (5.36)-(5.38)
- La valeur de la solution maître ré-évaluée sur les 500 premiers scénarios (en LP)
- La valeur de la solution maître ré-évaluée sur tout le jeu de données (en IP), c'est-à-dire sur 10 000 scénarios

La deuxième valeur permet de mesurer l'incidence de notre remplacement de la fonction valeur par sa minoration. Plus l'écart entre les première et deuxième valeurs est faible, plus notre heuristique sera précise. Afin de juger de la qualité des résultats obtenus, nous calculons d'autres valeurs :

- La valeur de la solution L-shaped continue sur 500 scénarios
- La valeur de cette dernière sur les 10 000 scénarios du jeu
- La valeur de la solution moyenne en tout LP
- La valeur de la solution moyenne en tout IP (sauf la sous-couverture)

La *valeur de la solution stochastique* (ou VSS) LP, c'est-à-dire le gain entre la solution moyenne et la solution L-shaped est déterminé ainsi de manière exacte. Il nous servira de gain de *référence*. Pour calculer le gain qu'apporte notre solution heuristique, on compare son coût réel en IP (c'est-à-dire évalué sur 10 000 scénarios) avec le coût réel de la solution moyenne en tout IP. En gardant à l'esprit que le gain IP n'est pas exact car on utilise une tolérance pour l'algorithme de branch and bound lors de l'évaluation du coût réel, on peut utiliser le gain LP pour juger de la qualité du gain IP.

Description des instances

Nous avons utilisé 10 différentes instances, décrites ci-dessous :

Standard : L'instance utilisée en premier lieu.

Double pic : déjà décrite.

Rand 2 : il s'agit de l'instance « grande variance » déjà présentée. La demande prévisionnelle est générée en prenant la demande « standard » et en ajoutant aléatoirement +2 ou -2 pour chaque période.

Rand 3 : Reprend la demande prévisionnelle standard (figure 5.1), avec un plus grand écart-type pour les scénarios générés (cf. tableau 5.9).

Rand 4 : Idem Rand 3, avec un écart-type plus grand encore.

Double pic 2 : Reprend la demande prévisionnelle de l'instance Double pic, avec une variance plus faible.

M7-M8-M9-M10 : Demande prévisionnelle standard, de moins en moins de variations « haute fréquence ».

Instance	STD	DPic	RAND 2	RAND 3	RAND 4
Ecart-type absolu moyen	1.13	12.90	13.00	6.34	8.31
Ecart-type relatif moyen (%)	6.1	68	68	34	45

Instance	DPic 2	M7	M8	M9	M10
Ecart-type absolu moyen	6.16	8.68	10.06	6.15	11.41
Ecart-type relatif moyen (%)	33	42	49	32	57

TABLEAU 5.9 Ecart-type des différences instances

Le tableau 5.10 résume les différentes tolérances utilisées pour les résolutions successives.

Description	Valeur
B&B final heuristique	0.1 %
B&B evalution IP	0.5 %
B&B solution moyenne IP	0.2 %

Tableau 5.10 Tolérances et paramètres utilisés

5.3 Résultats

5.3.1 Calcul des solutions moyennes

En plus des solutions moyennes en IP et en LP, nous avons calculé la valeur réelle des solutions L-shaped LP pour 500 scénarios. Ainsi, nous pouvons calculer la valeur de la solution stochastique en LP (c'est-à-dire le gain entre solution moyenne et solution L-shaped). Les résultats sont présentés dans le tableau 5.11.

Instance	STD	DPic	RAND 2	RAND 3	RAND 4
Valeur sol. moy. LP	453.992	611.440	631.978	472.903	491.551
Valeur sol. moy. IP	457.306	612.048	633.215	473.692	493.107
Gap d'intégrité (%)	0.73	0.10	0.20	0.17	0.32
VSS LP (%)	0.37	6.90	8.68	0.55	0.87

Instance	DPic 2	M7	M8	M9	M10
Valeur sol. moy. LP	481.794	477.263	528.427	492.099	609.242
Valeur sol. moy. IP	482.953	478.531	529.361	492.700	609.841
Gap d'intégrité (%)	0.24	0.27	0.18	0.12	0.10
VSS LP (%)	0.46	0.76	3.34	0.59	15.57

Tableau 5.11 Solutions moyennes et valeur de la solution stochastique LP

La dernière ligne de ce tableau est la plus importante : elle permet de juger l'intérêt (en LP) de calculer une solution stochastique plutôt que de calculer une solution moyenne. Pour les instances DPic et RAND 2, l'intérêt est notable, puisque des gains

de l'ordre de 7 à 8 % sont constatés. L'instance M10 mène à un gain encore plus important : 15%.

En outre, nous avons constaté qu'en IP comme en LP, les solutions moyennes étaient en général *multiples* : pour une même demande moyenne, il existe différentes allocations de quarts optimales. Le problème est que ces différentes solutions n'ont pas le même coût sur 10 000 scénarios (ce que nous avons appelé le *coût réel*). Cela montre justement une des faiblesses de ce genre de solutions.

Afin d'évaluer l'importance de ce phénomène, nous avons calculé, pour chaque instance, 20 solutions moyennes en introduisant, pour chaque quart j, une perturbation aléatoire $\epsilon_j \in [-10^{-5},\ 10^{-5}]$ de son coût, différente pour chaque évaluation.

Ainsi, les solutions au problème moyen ne seront plus multiples. On calcule ensuite le coût sur 10 000 scénarios de ces solutions. Les résultats, en relatif par rapport à la valeur réelle moyenne des solutions moyennes obtenues, sont présentés sur la figure 5.9. Les segments représentent la dispersion des coûts réels obtenus sur différentes instances (le coût moyen étant ramené à 100 pour facilier la comparaison). Le carré rouge représente le coût réel de la solution calculé grâce à l'heuristique (cf. paragraphes suivants).

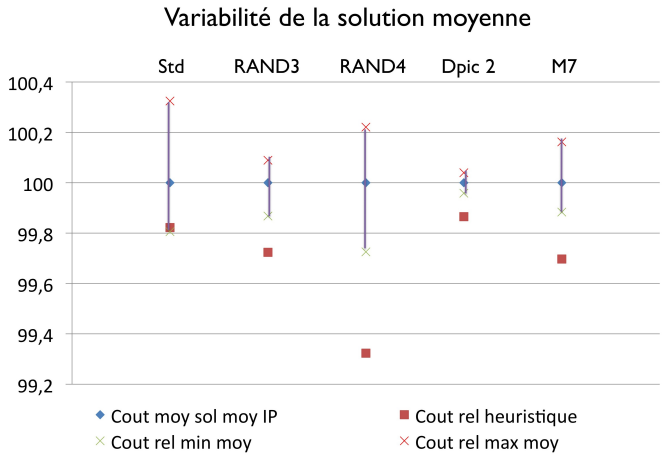


FIGURE 5.9 Variabilité des solutions moyennes

On constate que les variations des solutions moyennes sont largement supérieures à celles inhérentes aux perturbations introduites, qui sont de l'ordre de $10^{-6}\%$. Cela confirme bien qu'il existe un phénomène de solutions multiples. Dorénavant, nous calculerons un coût moyen sur 20 tirages afin de déterminer la valeur réelle de « la » solution moyenne. Il faudra garder à l'esprit que ce gain n'est qu'un gain *moyen*.

On remarque également que pour certaines instances où le gain moyen est très faible, il est parfois possible, si l'on a une « bonne » solution déterministe, de perdre de l'optimalité en utilisant une approche stochastique (*i.e.* avoir un gain négatif).

5.3.2 Calcul des solutions heuristiques

Nous avons effectué 9 séries de calculs afin de calculer les valeurs décrites précédemment. La figure 5.10 présente, en premier lieu, le temps nécessaire au calcul de la solution heuristique suivant les instances.

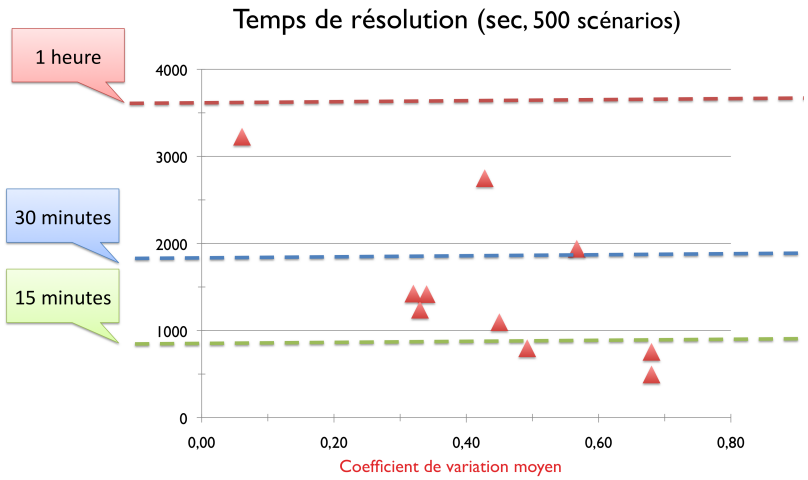


FIGURE 5.10 Temps de calcul

La majorité du temps est consacrée à la première étape de notre heuristique (L-shaped et fixation de variables), le temps pour l'algorithme de branch and bound final est de l'ordre de 20 secondes. Après divers essais, le seuil de fixation des variables à l'entier supérieur a été choisi à 0.8, car il présentait les meilleurs résultats de notre série de tests.

Bien que les temps de branchement soient courts, le faible gap d'intégrité constaté précédemment fait qu'il est difficile de diminuer la tolérance sans que le temps n'explose (1 heure minimum) sur une des instances au moins.

Le tableau 5.12 récapitule les résultats importants de cette série de calculs.

Instance	STD	DPic	RAND 2	RAND 3
Valeur LP 500 heuristique	451.276	577.234	591.110	480.303
Sous-estimation heuristique (%)	0.12	0.09	0.08	0.10
Gap L-shaped LP / Heur. IP (%)	0.18	0.13	0.14	0.16
Valeur réelle heuristique	456.509	570.6601	578.705	472.796
Valeur Sol. Sto. IP (%)	0.18	6.81	8.38	0.28
Fraction de gain capturée (%)	33.93	98.09	96.43	55.99

Instance	RAND 4	DPic 2	M7	M8	M9	M10
Val. LP 500 heur.	479.379	481.805	473.584	516.905	484.600	511.017
Sous-est. heur. (%)	0.08	0.18	0.10	0.12	0.002	0.10
Gap LP /IP (%)	0.12	0.27	0.13	0.17	0.0045	0.19
Val. heur.	490.002	482.568	475.927	513.258	489.975	516.398
VSS IP (%)	0.68	0.14	0.30	2.72	0.59	15.323
Gain capt. (%)	63.43	27.24	51.00	85.15	98.01	98.01

Tableau 5.12 Résultats finaux

La valeur LP de l'heuristique est la valeur de la solution maître trouvée, évaluée (en LP) sur les 500 premiers scénarios. Elle diffère de la solution « affichée » par l'heuristique, car la fonction recours a été approximée par des hyperplans minorants.

Cette différence est calculée dans la ligne « sous-estimation heuristique ». La valeur réelle heuristique est la valeur de la solution maître trouvée par l'heuristique, évaluée en IP sur les 10 000 scénarios de l'instance concernée.

Comme nous l'avons précisé en section 5.3.1, le gain par rapport à la solution moyenne (dite également *déterministe*) est calculé à partir d'une valeur *moyenne* des différentes valeurs des solutions déterministes obtenues en perturbant les coûts. La figure 5.11 présente, en pourcentage, la valeur de la solution stochastique en IP pour

nos différentes instances, c'est-à-dire la différence entre le coût réel de la solution heuristique et celui de la solution moyenne.

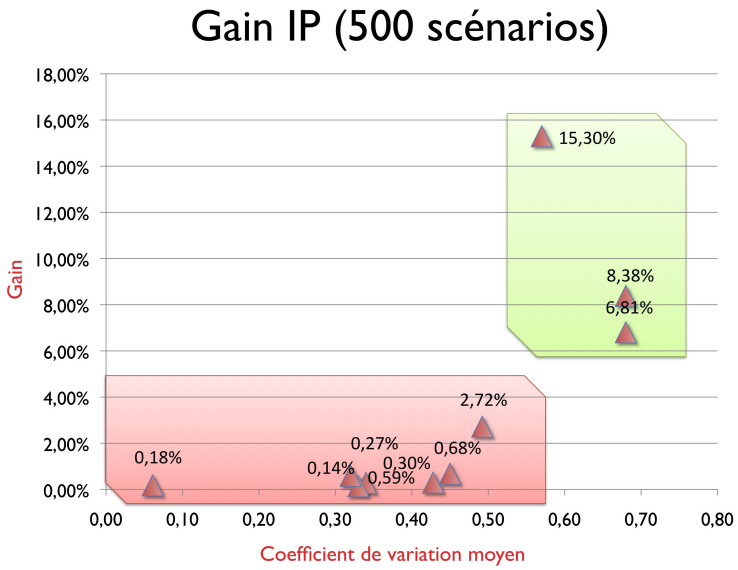


FIGURE 5.11 Gain obtenu en IP entre solution stochastique et solution moyenne

5.3.3 Comparaison et commentaires

La figure (5.12) résume graphiquement la fraction de gain LP capturé en IP, et constitue donc une mesure de la performance de notre heuristique. L'axe des abscisses correspond à l'écart-type relatif moyen du jeu de données, et l'axe des ordonnées correspond au pourcentage de gain LP capturé par notre résolution IP.

En conclusion de cette étude, on constate que bien que la solution stochastique est toujours meilleure que la solution « moyenne », la différence entre ces deux solutions ne justifie pas systématiquement que l'on s'attaque au problème stochastique, bien plus complexe à résoudre que le problème moyen. Cela est vrai en particulier lorsque le gain en LP se retrouve être inférieur à 1 % : l'imprécision de la résolution (approximations pour évaluer la valeur réelle, heuristique de résolution, solution moyenne dégénérée) fait que les résultats sont extrêmement variables d'une solution à l'autre, et peu satisfaisants.

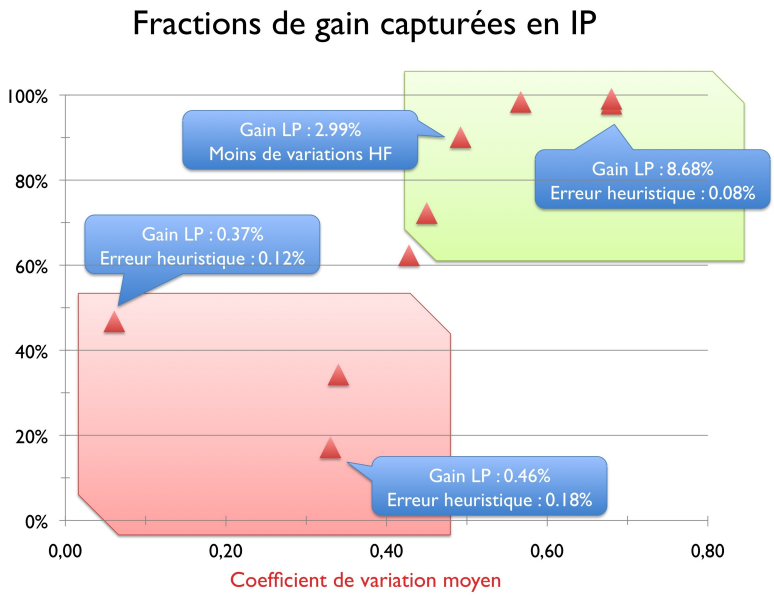


FIGURE 5.12 Représentation du gain capturé

En revanche, pour les demandes où la variance est élevée, notre méthode présente un réel intérêt. La méthode L-shaped que nous avons implémentée permet de résoudre en un temps environ 10 fois moindre que `CPLEX` le problème en continu, qui comporte environ 37 millions de variables. La ré-exploitation des coupes de Benders dans le processus heuristique de branchement, combiné à un faible gap d'intégrité, nous permet d'avoir accès rapidement à des solutions IP qui sont dans la plupart des cas à moins de 0.2% de l'optimum : c'est une précision considérée comme très bonne en pratique. De plus, le nombre important de scénarios qu'elle permet d'envisager est un gage de *stabilité* : la méthode dépend beaucoup moins des scénarios sélectionnés que si l'on n'en considérait que 25 par exemple.

Au final, notre méthode de résolution est à la fois rapide et précise, pour les problèmes où elle présente un intérêt. Elle est moins utile pour des problèmes où la demande varie peu. Pour ces derniers, les gaps d'intégrité et autres incertitudes annulent une grande partie du faible gain qu'il est possible de récupérer. Il doit néanmoins être possible pour ces dernières de diminuer le nombre de scénarios à prendre en compte afin de tenter d'améliorer la précision de l'heuristique.

Il reste désormais à tester notre méthode sur des données issues de la réalité, qui n'ont pas encore été fournies par l'entreprise associée à ce projet. Nous pourrons ainsi tester de manière plus poussée l'influence de la forme de la demande prévisionnelle, qui risque d'être beaucoup moins lisse que dans notre modèle : cela laisse optimiste quant aux améliorations que pourra apporter notre méthode.

Chapitre 6

CONCLUSION

6.1 Synthèse des travaux

L'essentiel de l'apport de nos travaux réside dans les trois points suivants :

1. Nous avons utilisé une approche novatrice pour l'optimisation d'horaires de personnel ;
2. Nous avons développé une heuristique rapide et efficace de résolution d'un problème MIP comportant à l'origine 10 millions de variables ;
3. Nous avons évalué quantitativement l'intérêt d'une approche stochastique par rapport à une approche déterministe.

Décrivons plus précisément chacun de ces points :

Approche stochastique : La plupart des problème d'optimisation d'horaires avec demande variable utilisent l'approche déteministe, car très rapide à mettre en place. Nous nous sommes, nous, intéressés à une formulation stochastique du problème. Cette approche est novatrice, puisque d'une part le domaine des problèmes stochastiques en nombres entiers de grande taille a été peu exploré jusque là, et d'autre part, leur application sur le problème de construction de quarts à grande échelle et avec un grand nombre de scénarios n'a jamais été abordé. Les résultats obtenus montrent qu'il est intéressant de poursuivre dans cette voie.

Développement d'une heuristique : Après avoir adapté le problème classique de construction de quarts au cas stochastique, nous avons développé une heuristique de résolution adaptée au problème. Cette dernière offre des résultats satisfaisants à la fois en termes de précision (voir par exemple la section 5.2.2, consacrée à l'étude d'un modèle « jouet »), et également en terme de rapidité. Ainsi, alors qu'une résolution en IP par `CPLEX` demandait plus de 7 000 secondes pour 25 scénarios et une précision de

0.2%, notre heuristique demande en général moins d'une demi-heure de calcul pour résoudre le problème à 500 scénarios, avec une précision meilleure (mais non garantie).

Intérêt d'une approche stochastique : Nous avons également évalué numériquement l'intérêt d'une approche stochastique, plus lourde à mettre en place qu'une résolution déterministe. Dans des cas où la variance entre les scénarios est faible, une approche stochastique a peu d'intérêt. Pire : si l'on obtient une « bonne » solution déterministe, il est même possible que la solution moyenne obtenue soit en réalité plus chère que cette solution déterministe. En revanche, dès que la variance est suffisamment élevée, on constate des gains intéressants avec notre méthode (de 3 à 8 voire 15 % suivant les cas de figure). Dans tous les cas, la solution stochastique apporte une stabilité en terme de coût, tandis que l'on n'a a priori aucun contrôle sur le coût des solutions moyennes, qui varie dans nos instances de l'ordre de 0.5%.

6.2 Limitations de la solution proposée

La principale limitation de notre méthode est que nous n'avons pas eu l'occasion, pour l'instant, de la tester sur des données issues de la réalité. Même si nous avons essayé de la reproduire au mieux, grâce entre autres aux travaux de Legrain (2011) dans ce domaine et en essayant diverses formes de demande prévisionnelle, nous ne pouvons pas prédire exactement ce qu'il va se passer en réalité.

En outre, même si les gains apportés en théorie par la méthode sont intéressants, qu'en sera-t-il en pratique ? Quel est l'intérêt d'une solution présentant un gain théorique de 3% si les erreurs de mesure ou d'exécution sont du même ordre ? Un employé ne sera jamais précis à la seconde près.

Pour les deux raisons évoquées ci-dessus, il est important de pouvoir évaluer l'intérêt réel de notre approche, afin de comparer les gains théoriques aux gains en pratique.

6.3 Améliorations futures

Pour conclure ce rapport, nous présentons quelques pistes qu'il serait intéressant d'explorer.

Estimation du gain : Tout d'abord, il faudrait déterminer les critères précis qui font qu'un jeu de données va donner de bons résultats ou non : forme de la demande prévisionnelle, variance, durée et intensité des perturbations,...Ainsi, il suffirait qu'une entreprise nous fournisse son propre jeu de données pour qu'en quelques secondes et sans résoudre le problème, on puisse lui dire s'il est intéressant pour elle de faire appel à une approche stochastique ou non.

Par exemple, une inégalité classique en programmation stochastique donne :

$$VSS \leq EEV - EV \tag{6.1}$$

Cela signifie que le gain entre solution stochastique et solution moyenne est majoré par la différence entre le coût espéré, sur tous les scénarios, de la solution moyenne et le coût de la solution moyenne sur le scénario moyen. D'autres inégalités similaires existent. Peut-être que certaines d'entre-elles pourraient nous permettre de majorer rapidement et de manière assez fine ce gain.

En outre, on pourrait s'intéresser plus précisément au coût de la sous-couverture. On pourrait par exemple modéliser des phénomènes d'attente (une sous-couverture qui se « résorberait » immédiatement ne coûterait que très peu), d'impatience des clients, dots Au lieu de minimiser à tout prix la sous-couverture, le but serait plutôt de minimiser le nombre de clients non satisfaits, l'idée principale étant qu'il est en général moins grave de faire attendre 60 clients pendant 1 minute que de faire attendre 1 client pendant 60 minutes.

Amélioration de la résolution : Il est également possible de diminuer de manière sensible le temps de résolution du problème. Des mesures nous ont montré que les temps passés sur le problème maître et sur les problèmes secondaires sont à peu près équivalents lors de la méthode L-shaped. Nous avons deux propositions pour réduire le temps de résolution :

Problème maître : au départ très rapide, le problème s'alourdit vite (à raison de 500 coupes par itération de la méthode L-shaped, on arrive vite à 50 000 coupes). Ces coupes étant le seul facteur d'alourdissement du problème, on

pourrait songer à en supprimer certaines afin de retrouver un problème plus léger. La fonctionnalité de *contraintes dynamiques* dont est doté `CPLEX`, qui consiste à omettre des contraintes lors de la résolution et de seulement vérifier a posteriori qu'elles sont satisfaites pourrait être mise à profit. Il resterait à trouver un bon critère d'élimination des coupes « inutiles ».

Problème esclave : Tous les problèmes esclaves sont indépendants. Il est donc possible de les répartir sur plusieurs processeurs sans problème, et sans introduire de facteur aléatoire lors de la résolution. Nous avons déjà implémenté cette modification à l'algorithme, à raison de 125 scénarios par processeur, sur 4 processeurs. L'implémentation est aisée car `Java` facilite la gestion du multi-threading. Cela divise environ par 3 le temps pris par la résolution des problèmes esclaves. Dans les instances où le problème maître était rapide à résoudre, cela divisait par 2 le temps de résolution.

D'autres instants de décision : Enfin, il serait possible de mener une étude plus théorique sur l'intérêt des approches à n temps de décision. Est-il intéressant d'augmenter le nombre d'instants de décision (par exemple décider des quarts réguliers un mois avant, des temps partiels un jour avant avec des prévisions améliorées, et du reste en temps réel) ? L'approximation en deux étapes est-elle suffisante ?

Références

AHMED, S. et GARCIA, R. (2004). Dynamic capacity acquisition and assignment under uncertainty. *Annals of Operations Research*, 124, 267–283.

AHMED, S., TAWARMALANI, M. et SAHINIDIS, N. (2004). A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming A*, 100, 355–377.

AYKIN, T. (1996). Optimal shift scheduling with multiple break windows. *Management Science*, 42, 591–602.

BAPTISTE, P., GIARD, V., HAÏT, A. et SOUMIS, F. (2005). *Gestion de production et ressources humaines*, Presses internationales Polytechnique, chapitre 4 : Gestion des horaires et affectation de personnel.

BARD, J. F., MORTON, D. P. et WANG, Y. M. (2007). Workforce planning at USPS mail processing and distribution centers using stochastic optimization. *Annals of Operations Research*, 155, 51–78.

BECHTOLD, S. et JACOBS, L. W. (1990). Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science*, 36, 1339–1351.

BENDERS, J.-F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4, 238–252.

BIRGE, J.-R. et LOUVEAUX, F. (1997). *Introduction to stochastic programming*. Springer.

CAROE, C.-C. et SCHULTZ, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24, 37–45.

DANTZIG, G. (1954). A comment on Edie's "traffic delays at toll booths". *Journal of the Operations Research Scoiety of America*, 2, 339–341.

GRAVER, J. (1975). On the foundations of linear and integer linear programming I. *Mathematical Programming*, 8, 207–226.

HEMMECKE, R. et SCHULTZ, R. (2003). Decomposition of test sets in stochastic integer programming. *Mathematical Programming B*, 94, 323–341.

KONG, N., SCHAEFER, A. et HUNSAKER, B. (2006). Two-stage integer programs with stochastic right-hand sides : a superadditive dual approach. *Mathematical Programming B*, 108, 275–296.

LEGRAIN, A. (2011). *Génération de scénarios pour la demande en personnels durant plusieurs périodes*. Mémoire de maîtrise, École Polytechnique de Montréal.

MAGNANTI, T. et WONG, R. (1981). Accelerating Benders decomposition : Algorithmic enhancement and model selection criteria. *Operations Research*, 29, 464–484.

PAPADAKOS, N. (2008). Practical enhancements to the Magnanti-Wong method. *Operations Research Letters*, 36, 444–449.

SCHULTZ, R., STOUGIE, L. et VAN DER VLERK, M. (1998). Solving stochastic programs with integer recourse by enumeration : A framework using Gröbner basis reductions. *Mathematical Programming B*, 83, 229–252.

SHERALI, H. et ZHU, X. (2009). *Advances in Applied Mathematics and Global Optimization*, Springer Science, chapitre 12 : Two-Stage Stochastic Mixed-Integer Programs : Algorithms and Insights.

VAN SLYKE, R. et WETS, R. (1969). L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM Journal on Applied Mathematics*, 17, 638–663.